Foxit

User Manual

Foxit® PDF Toolkit

Microsoft® Partner
Gold Independent Software Vendor (ISV)

Table of Contents

1 Introduction to Foxit PDF Toolkit

Foxit PDF Toolkit consists of a series of command line tools to perform batch PDF generation and processing. It includes ten modules: Image2PDF, Office2PDF, PDFWatermark, PDFHeaderFooter, PDFOptimizer, PDFRedactor, PDFMetadata, PDF2Text, Text2PDF and RMS. Each module offers a simple-to-use API for users who want to perform PDF manipulation through API.

In this manual, we will introduce the modules: Image2PDF, Office2PDF, PDFWatermark, PDFHeaderFooter, PDFOptimizer, PDFRedactor, PDFMetadata, PDF2Text and Text2PDF. For the RMS PDF Protection module, the user manual will cover in detail the module's basic syntax in section 3.10 "RMS". Browse to the "rms" folder in the installation package for a more detailed introduction.

Image2PDF is an easy-to-use command line tool used to batch convert large volumes of image files into high-quality PDF files, without needing to install any additional software. Image2PDF currently supports the following image formats: BMP, PNG, JPEG, JPX, GIF, TIFF (or TIF, including the image with a single page or multiple pages). This module provides many features to customize the output properties of the generated PDF files.

Office2PDF batch converts Microsoft Office documents into professional-quality PDF files. Microsoft Office must be installed, because the Office2PDF tool saves the converted PDF files using the Microsoft Office Engine. Office2PDF currently supports the more popular document formats, which includes doc, docx, xls, xlsx, ppt and pptx. Additionally, the tool can convert and create professional-quality PDF files that support PDF/A Standard compliances.

PDFWatermark allows users to batch add a watermark into PDF files, without needing to install any additional software. The PDFWatermark tool applies a pre-made watermark to PDF files, which supports watermarks in both text or image formats. The watermark is saved as a configuration file with the extension ".xml", which is generated by the build-in Foxit Configuration Tool (Either *fpdfwmconf.exe* or *fpdfwmconf64.exe*, a GUI application, depending on your operating system).

PDFWatermark allows users to batch add a watermark into PDF files, without needing to install any additional software. The PDFWatermark tool applies a pre-made watermark to PDF files, which supports three types of watermarks: text, image and PDF. The watermark is saved as a configuration file with the extension ".xml", which is generated by the build-in Foxit Configuration Tool (Either *fpdfwmconf.exe* or *fpdfwmconf64.exe*, a GUI application, depending on your operating system).

PDFHeaderFooter batch adds header/footer into large volumes of PDF files, without needing to install any additional software. The PDFHeaderFooter tool applies pre-made header/footer to PDF files. The headers and footers are saved as a configuration file with the extension ".xml", which is generated by the build-in Foxit Configuration Tool (Either *fpdfhfconf.exe* or *fpdfhfconf64.exe, a GUI application,* depending on your operating system).

PDFOptimizer batch reduces the size of PDF files to reduce disk space, making them easier to share or archive, without needing to install any additional software. This tool optimizes PDF files by downsampling or compressing images, unembedding fonts, and discarding objects or user data. PDFOptimizer offers some options to clean up PDF files, such as using Flate to encode streams that are not encoded, converting LZW encoding to Flate encoding, and removing invalid bookmarks and invalid links.

PDFRedactor batch removes sensitive content or private information from your PDF documents prior to making them available to others, without needing to install any additional software. With PDFRedactor, users can use redaction marks to redact or remove the sensitive contents that are visible in a PDF document and can specify custom text to appear over the redaction marks.

PDFMetadata batch adds document metadata information to PDF files, without needing to install any additional software. With PDFMetadata, users can set the title, author, subject, keywords and creator of PDF files, and view the metadata information.

PDF2text batch converts large volumes of PDF files into plain text files, without needing to install any additional software. With PDF2Text, users can extract text content from a textual PDF file and save the text as a plain text file, which is useful for text indexing or content retrieval.

Text2PDF batch converts large volumes of plain text files into high-quality PDF files with a minimum loss of formatting information, without needing to install any additional software. This tool converts plain text to PDF files including settings of the page size, page margin, font style, font size, font color, password, and metadata.

1.1 Why Foxit PDF Toolkit is your choice

Foxit is an Amazon-invested leading software provider of solutions for reading, editing, creating, organizing, and securing PDF documents. Foxit PDF Toolkit is a suite of modules to perform high volume PDF creation and processing on Windows servers. These modules help IT organizations develop workflows to process large amounts of PDF files. They also provide libraries for software development groups to incorporate PDF processing into their applications. Customers choose this product for the following reasons:

- **High performance** – Multi-threading support speeds up the PDF processing based on today's server architectures.
- **Professional quality** – Professional-quality on PDF manipulation and PDF conversion.
- **Lightweight footprint** – Lower memory usage and faster installation.
- **Perfect message mechanism** – Gives more perfect message hints if users encounter problems when using the tools, so Foxit PDF Toolkit can offer better user experience.
- **Robust and stable** – Ensures smooth running of the application and enhancement of fault tolerant.
- **Easy to integrate** – Command line or application interfaces enable flexible and seamless integration with user's existing workflows.
- **Plug and Play** – Choose one or more of the specific modules that meet your needs.

Foxit offers 24/7 support for its products and are fully supported by the PDF industry's largest development team of support engineers. Updates are released on a regular basis to improve user experience by fixing bugs and adding new features and functionalities. Foxit PDF Toolkit is the best solution for PDF batch processing and integration of high performance features at a low cost!

1.2 Foxit PDF Toolkit Features

1.2.1 Image2PDF Features

- Batch convert image files into PDF files in server environments.
- Support multi-thread processing.
- Support sub-folder recursion processing.
- Support single-file processing, single-folder processing and multi-file processing.
- Support various image formats like BMP, PNG, JPEG, JPX, GIF and TIFF (TIF).
- Convert multiple image files into one PDF file, or individual PDF files.
- Specify any resolution (DPI) for the PDF file to be converted.
- Set page width and height for the PDF file to be converted.
- Set open password for the PDF file to be converted.
- Add bookmarks to the PDF file to be converted.
- Set margin for each PDF page.
- Set document metadata information of PDF files, such as the title, subject, keywords, author and creator.
- Support wildcard character in batch conversion, e.g., *.jpg, *.png.
- Offer simple-to-use API.

1.2.2 Office2PDF Features

- Batch convert Microsoft Office files into PDF files in server environments.
- Support multi-thread processing.
- Support sub-folder recursion processing.
- Support single PDF file processing or single-folder processing.
- Support password-protected Microsoft Office files.
- Support the popular Microsoft Office document formats, including doc, docx, xls, xlsx, ppt and pptx.
- Add bookmarks to PDF files converted only from Microsoft Word documents.
- Support scale options for Microsoft Excel conversion.
- Convert every sheet of a Microsoft Excel document into a single PDF file.
- Retain hyperlinks in PDF files from Microsoft Office files.
- Support wildcard character in batch conversion, e.g., *.docx, *.pptx.
- Support PDF/A Standard for the PDF file to be converted.
- Offer simple-to-use API.

1.2.3 PDFWatermark Features

- Batch add a watermark into PDF files.
- Support multi-thread processing.
- Support sub-folder recursion processing.
- Support single PDF file processing or single-folder processing.
- Support password-protected PDF files.
- Support text, image, and PDF watermarks.
- Support stylized text watermarks that include font, font size, font color and underline settings.
- Support image watermarks in various formats (BMP, DIB, JPG, JPEG, JPE, GIF, TIF, PNG, and TIFF).
- Support one page and multi-page PDF watermarks.
- Rotate watermark in 360 degree rotations.
- Set opacity for watermarks.
- Scale relative to target page for watermarks.
- Control the visibility of watermarks when printing.
- Control the visibility of watermarks when being displayed on screen.
- Keep watermark position and size for different page sizes.
- Appear on top or in the background of the page.
- Set watermark position in PDF page.
- Specify a page range to add watermark.
- Set document metadata information of PDF files, such as the title, subject, keywords, author and creator.

- Support wildcard character in batch processing, e.g., *.pdf.
- Preview watermark in the build-in Foxit Configuration Tool.
- Offer simple-to-use API.

1.2.4 PDFHeaderFooter Features

- Batch add header/footer into PDF files.
- Support multi-thread processing.
- Support sub-folder recursion processing.
- Support single PDF file processing or single-folder processing.
- Support password-protected PDF files.
- Support stylized text for the header/footer, including font, font size, font color, underline and embedded font settings.
- Support margins settings.
- Shrink a document to avoid overwriting the document's text and graphics.
- Keep position and size of header/footer constant on different page sizes.
- Set page number and date as header/footer.
- Support page number and date format settings.
- Specify the page range to add header/footer.
- Set document metadata information of PDF files, such as the title, subject, keywords, author and creator.
- Support wildcard character in batch processing, e.g., *.pdf.
- Preview header/footer in the build-in Foxit Configuration Tool.
- Offer simple-to-use API.

1.2.5 PDFOptimizer Features

- Batch optimize PDF files.
- Support multi-thread processing.
- Support sub-folder recursion processing.
- Support single PDF file processing or single-folder processing.
- Support password-protected PDF files.
- Support downsampling and compression for color/grayscale and monochrome images.
- Unembed fonts in PDF files.
- Discard objects and user data of PDF files:
    - Discard all form submission, import and reset actions.
    - Flatten form fields.
    - Discard all JavaScript actions.
    - Discard embedded page thumbnails.

- Discard embedded print settings.
- Discard bookmarks.
- Discard all comments, forms and multimedia.
- Discard external cross references.
- Discard document information and metadata.
- Discard file attachments.
- Discard private data of other applications.

- Clear up PDF files:
  - Discard all form submission, import and reset actions.
  - Use Flate to encode streams that are not encoded.
  - In streams that use LZW encoding, use Flate instead.
  - Remove invalid bookmarks.
  - Remove invalid links.
- Set document metadata information of PDF files, such as the title, subject, keywords, author and creator.
- Support wildcard character in batch processing, e.g., *.pdf.
- Offer simple-to-use API.

1.2.6 PDFRedactor Features

- Batch redact or remove sensitive content from PDF files.
- Support multi-thread processing.
- Support sub-folder recursion processing.
- Support single PDF file processing or single-folder processing.
- Support password-protected PDF files.
- Option to set the search keyword mode for specific words/phrases or patterns.
- Specify the characters of keyword you want to redact.
- Set a page range to apply redaction.
- Set a fill color to mark redaction.
- Overlay redaction marks with specified text.
- Set the alignment of the overlay text.
- Set the font style of the overlay text.
- Set the font size of the overlay text.
- Set the font color of the overlay text.
- Set document metadata information of PDF files, such as the title, subject, keywords, author and creator.
- Support wildcard character in batch processing, e.g., *.pdf.
- Offer simple-to-use API.

1.2.7 PDFMetadata Features

- Batch add document metadata information to PDF files.
- Support multi-thread processing.
- Support sub-folder recursion processing.
- Support single PDF file processing or single-folder processing.
- Support password-protected PDF files.
- Set the title of PDF files.
- Set the subject of PDF files.
- Set the keywords of PDF files.
- Set the author of PDF files.
- Set the creator of PDF files.
- View PDF metadata information.
- Support wildcard characters in batch processing, e.g., *.pdf.
- Offer simple-to-use API.

1.2.8 PDF2text Features

- Batch convert PDF files into plain text files.
- Support multi-thread processing.
- Support sub-folder recursion processing.
- Support single-file processing and single-folder processing.
- Support password-protected PDF files.
- Specify a page range you want to extract text from.
- Get the total number of characters on each page.
- Print the characters number of each page to the screen.
- Option to retain PDF page layout.
- Option to retain the page break of PDFs.
- Option to convert each PDF page into individual text files.
- Support UTF8 and UTF16 text encoding.
- Support wildcard character in batch processing, e.g., *.pdf.
- Offer simple-to-use API.

1.2.9 Text2PDF Features

- Batch convert text files into PDF files.
- Support multi-thread processing.
- Support sub-folder recursion processing.
- Support single-file processing and single-folder processing.

- Support most commonly used languages.
- Set page width and height for the PDF file to be converted.
- Set margin for each PDF page.
- Support font style, font size and font color settings.
- Set open password for the PDF file to be converted.
- Set document metadata information of PDF files, such as the title, subject, keywords, author and creator.
- Support wildcard character in batch processing, e.g., *.txt.
- Offer simple-to-use API.

1.3 Common Use Case Scenarios

1.3.1 Image2PDF Use Case Scenarios

- Batch convert images into PDF files in server environments for better image management.
- Create electronic books for users to scan paper documents to image files, and then convert them into a single PDF file.
- Make image albums for users to collect their photos and then convert them into a single PDF file.

1.3.2 Office2PDF Use Case Scenarios

- Batch convert Microsoft Office files into PDF files in server environments.
- Set the converted PDF files to be compliant with PDF/A standard for better archiving.

1.3.3 PDFWatermark Use Case Scenarios

- Batch add a watermark into PDF files to protect users and their copyrights. Users can use company logos, author signatures, products, PDF files or web addresses as watermarks to protect their PDF files.
- Label PDF file status for better management. Users can easily label status of their PDF files, such as approved, draft, final or confidential. For example, label “Confidential” on PDF pages that include sensitive information or “Draft” on a preliminary PDF document to be distributed for review.

1.3.4 PDFHeaderFooter Use Case Scenarios

- Batch add header/footer into PDF files to include information such as the date, page numbers, or the title of the document.

1.3.5 PDFOptimizer Use Case Scenarios

- Batch optimize large PDF files for electronic document exchange or space-saving document archiving in server environments. Large PDF files are not suitable for electronic document exchange or document archiving. Users can optimize large PDF documents instead of converting them into other formats.

1.3.6 PDFRedactor Use Case Scenarios

- Batch redact or remove sensitive content or private information from PDF files. Users can redact or remove sensitive text that are visible in PDF documents before distributing them to others, which could also help prevent the receivers from tracing the documents back to the sender.

1.3.7 PDFMetadata Use Case Scenarios

- Batch add document metadata information to PDF files. Users can set the title, author, subject, keywords and creator of PDF files, which could help them to find particular documents through searching these descriptions. For example, the keywords can be particularly useful for narrowing searches.

1.3.8 PDF2Text Use Case Scenarios

- Batch convert PDF files into plain text files for better archiving and indexing PDF files. It is indeed helpful for text indexing and content retrieval by creating a full-text searchable archive database.

1.3.9 Text2PDF Use Case Scenarios

- Batch convert plain text files into PDF files for better viewing or editing. It is not easy for users to view or edit plain text files, and in this case, users can convert them into PDF files for further processing.

1.4 System Requirements

Windows Platform：

- Windows Server 2012
- Windows Server 2008 R2
- Windows Server 2003

- Windows 10
- Windows 8.1
- Windows 7
- Windows Vista

Note *For the Office2PDF tool, please make sure that you have already installed Microsoft Office 2007 SP2 or later, and the virtual printers. If Microsoft Office 2007 is not the SP2 version, please download the "Microsoft Save as PDF" plugin from http://www.microsoft.com/en-us/download/details.aspx?id=9943.*

1.5 About This Manual

This manual aims at introducing the command line usage of Foxit PDF Toolkit. It is intended for the audiences who want to batch generate or process PDF files in server environments.

- Section 1: gives an introduction of Foxit PDF Toolkit.
- Section 2: illustrates how to install and uninstall Foxit PDF Toolkit.
- Section 3: describes basic usage of Foxit PDF Toolkit.
- Section 4: introduces Foxit Configuration Tool of PDFWatermark and PDFHeaderFooter modules.
- Section 5: shows how to work with API.
- Section 6: provides support information.

2 Installing and Uninstalling Foxit PDF Toolkit

2.1 Installation

Installation of Foxit PDF Toolkit is straightforward. You can download the trial release package, which is a zip file from our website (www.foxitsoftware.com), and then extract the package to the desired location as shown in the following figure. In this manual, we rename the package "pdftools" and unzip it to the directory "C:\pdftools". The package contains:

configtool: configuration tools for pdfwatermark and pdfheaderFooter

doc: user manual for Foxit PDF Toolkit

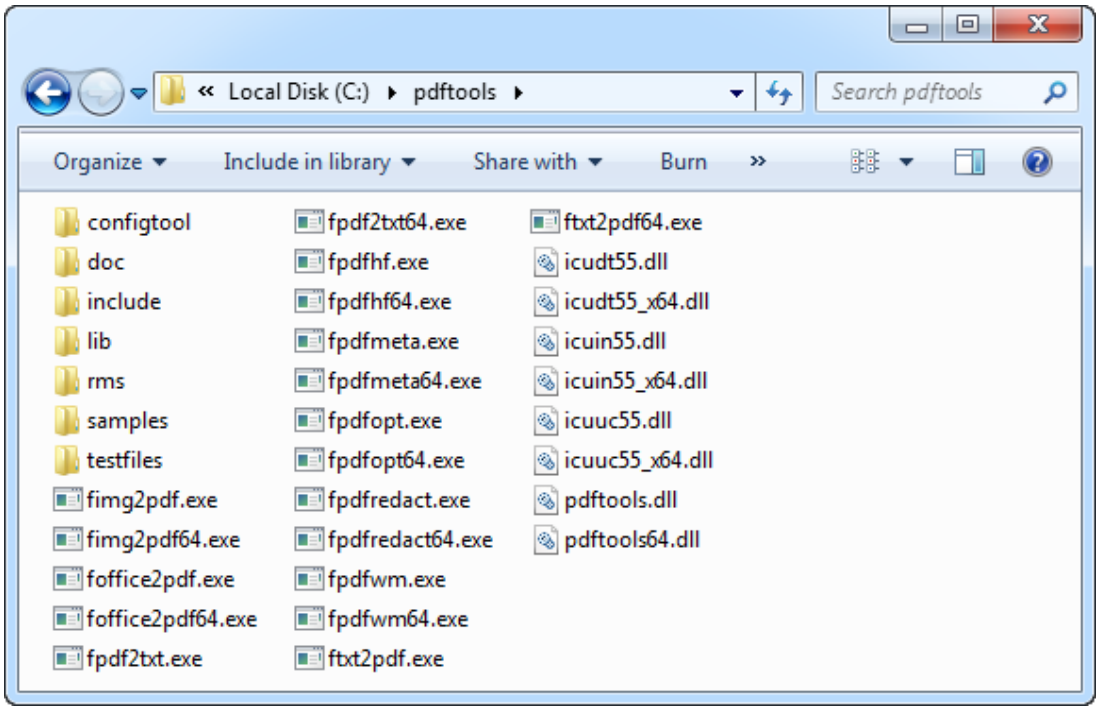
include: header file for Foxit PDF Toolkit

lib: third-part libraries and databases for Foxit PDF Toolkit

rms: RMS command line tool

samples: some batch samples for Foxit PDF Toolkit

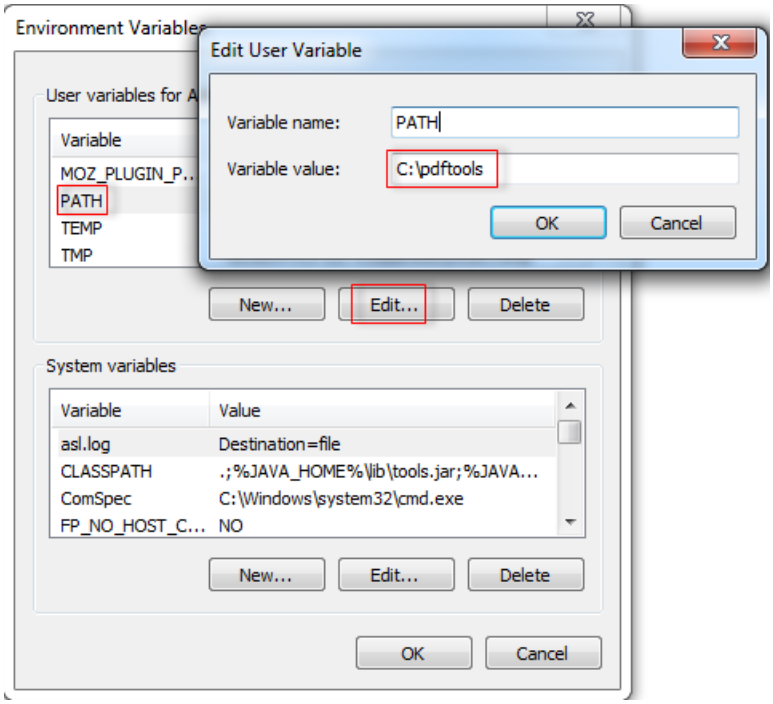
testfiles: testfiles for Foxit PDF Toolkit



Foxit PDF Toolkit package

After that, open a terminal session and change the directory to the installation location to run your application (except for PDFWatermark and PDFHeaderFooter modules, which are used only after using the Foxit Configuration Tool to generate configuration file containing the setting information of watermark or header/footer). For the Office2PDF module, please make sure that you have already installed Microsoft Office 2007 SP2 or later, and the virtual printers. If Microsoft Office 2007 is not the SP2 version, please download the "Microsoft Save as PDF" plugin from http://www.microsoft.com/en-us/download/details.aspx?id=9943.

Tips: You can set the installation path to Environment Variables, which allows you to use the commands in the terminal window directly without needing to change the directory to the installation location. Go to **Start**-> **Control Panel**-> **System**-> **Advanced system settings** -> **Advanced** -> **Environment Variables**, in the "User variables for Administrator" box, select **PATH** and **Edit**, and then add the installation path as shown in the following figure. If the environment variable "path" does not exist, create it by clicking **New**.



Set installation path to Environment Variables

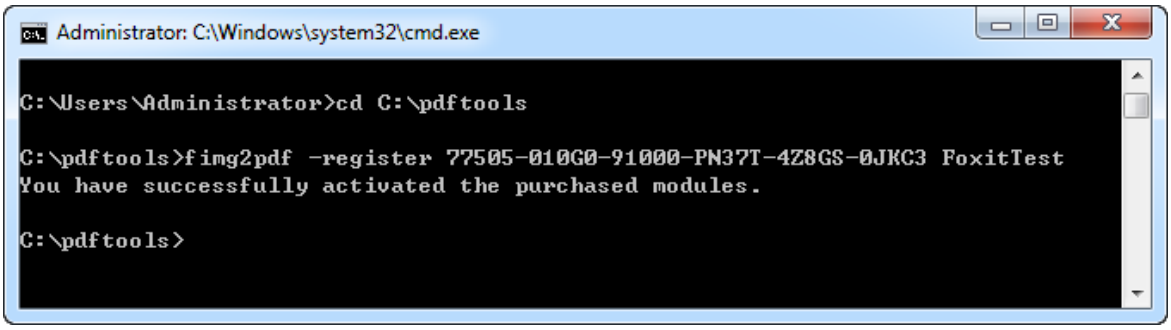
2.2 Evaluation

The Foxit PDF Toolkit is distributed on a "try-before-you -buy" basis. Foxit allows users to download trial version to evaluate the features. You have a free 30-day trial, but the pages of all created PDF files will contain our watermark, and for the PDF2Text module, you can only convert first ten PDF pages into plain text

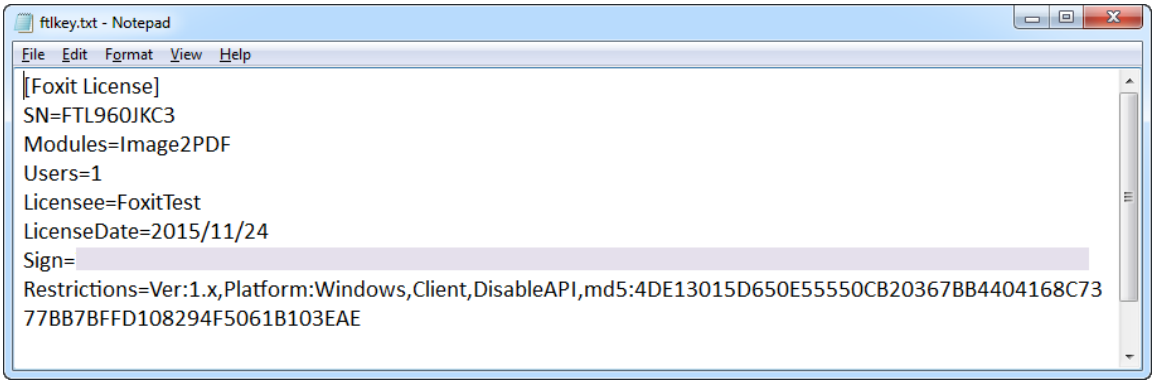
files. After the evaluation period, customers that want to continue to use the product should contact Foxit sales team to purchase corresponding activation codes for the desired modules.

2.3 Registration

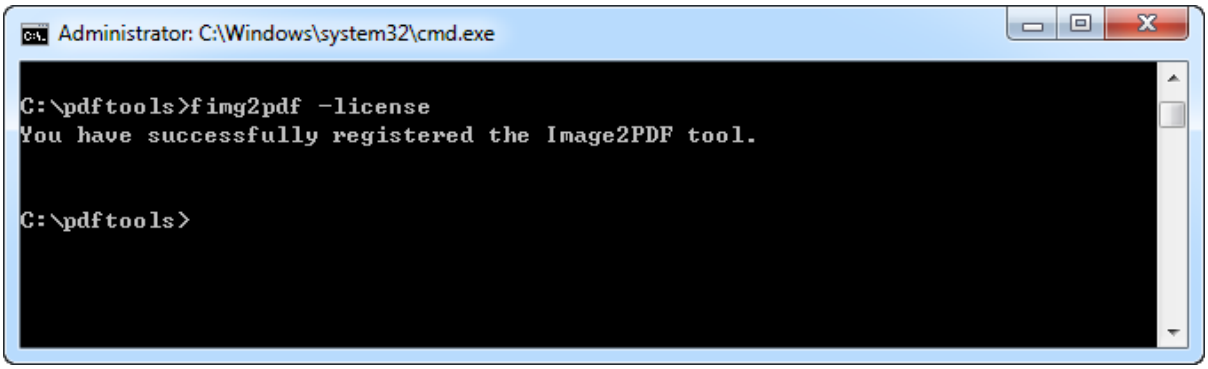
When you get the activation code for the module you want to purchase, please use the argument "**-register <code> <licensee>**" to active it in the command line window. For example, assume you get the activation code (77505-010G0-91000-PN37T-4Z8GS-0JKC3) for the Image2PDF module, you may type "fimg2pdf -register 77505-010G0-91000-PN37T-4Z8GS-0JKC3 FoxitTest" in the command line window as shown below.



Here, "FoxitTest" is the name of the licensee you designated. After activation, a key file named "**ftlkey.txt**" will be generated in the installed path with the contents shown in the following figure.



Then you can run "-fimg2pdf -license" in the command line window to check the license agreement as follows.



2.4 About License

Foxit PDF Toolkit provides three types of licenses for customers to choose.

- **Desktop License**- It is only available on desktop systems, which is perfect for personal or small business use. Each license is good for one user on one machine.
- **Server License**- It is available on servers, with 8 CPUs, which is good for small-to medium-sized businesses that need higher performance on a single server. If your server has more than 8 CPUs, please contact Foxit sales team to purchase Enterprise License.
- **Enterprise License**- It is intended for large companies that need to process a large number of PDF documents on multiple high-performance servers. Enterprise License also includes features for integrating the Foxit PDF Toolkit into your own applications. Please contact Foxit sales team to purchase Enterprise License.

2.5 Uninstallation

If you want to uninstall the Foxit PDF Toolkit, all you need to do is to delete the installed folder.

3 Command Line Usage

3.1 Image2PDF

3.1.1 Basic Syntax

fimg2pdf *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> [[-width <PDF width>] [-height <PDF height>]]*
[-dpi <resolution>] [-margin <left [top right bottom]>] [-b] [-sp <password>]
[-title <title>] [-subject <subject>] [-keywords <keywords>] [-author <author>] [-creator <creator>]
[-r [recursion]] [-t <thread>] [-log <logfile>] [-l <level>]

fimg2pdf *-register <code> <licensee>*

fimg2pdf *-license*

fimg2pdf *-version/-v*

fimg2pdf *-help/-h*

Note:

- <> required
- [ ] optional
- / mutually exclusive
- A space is needed between the command line argument and the value

Only the <*-i <srcfile/srcfolder>*> and <*-o <destfile/destfolder>*> arguments are actually required. All others are optional, which are available for controlling the output PDF files as desired. The arguments could be given in any order. Full details on each are explained in the following section.

3.1.2 Command Line Summary

Note For some arguments whose values are strings, users can choose whether to add *quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i c:\input\1.jpg* *-i "c:\input\1.jpg" "c:\ input\2.tif"*	Specifies the input file to be converted. ▪ The input string can be the name of a single image file (.bmp, .png, .jpg, .jpx, .gif, .tif, .tiff), multiple image files, or a folder.

Option	Parameter	Description
	-i c:\input *-i "c:\input*.jpg"*	▪ The file name can contain the wildcard character (*). For example, use *.tiff to include all TIFF image files in a given folder. **Note** Wildcard character (*.*) is currently not supported.
-o	*<-o <string>>* *e.g.* *-o d:\output\one.pdf* *-o d:\output*	Specifies the path of the output PDF file or folder. ▪ If user specifies a path of a PDF file, (e.g. *-o d:\output\one.pdf*), all input image files will be combined into a single PDF file. ▪ If user specifies a path of a folder, (e.g. *-o d:\output*), every input image file will be converted into individual PDF files. **Note** The specified output path must already exist.
-width	*[-width <points>]* *e.g.* *-width 612*	Sets the page width of the output PDF file in points. Default: Width of input image. **Note** The **-width** and **-height** options must be used together with a set value greater than 0.
-height	*[-height <points>]* *e.g.* *-height 792*	Sets the page height of the output PDF file in points. Default: Height of input image. **Note** The **-width** and **-height** options must be used together with a set value greater than 0.
-dpi	*[-dpi <integer>]* *e.g.* *-dpi 0* *-dpi 1* *-dpi 300*	Specifies DPI (Dots Per Inch) resolution of the output PDF file. The default value is 72. • **-dpi 0**: uses the default image width and height information. • **-dpi 1**: uses the DPI value of the input image. • **-dpi <integer>**: sets the DPI resolution as the given integer. **Note** ▪ If the program failed to get the DPI value of the input image when the (**-dpi**) is set to 1, the DPI value will be 96. ▪ The -weight, -height and -dpi options should not be used together.

Option	Parameter	Description
		♦ If you only use -width and -dpi, or -height and -dpi, the width or height setting will be omitted. ♦ If you use -width, -height and -dpi, the dpi setting will be omitted.
-margin	*[-margin <points [points points points]>]* *-margin <left [top right bottom]>* *e.g.* *-margin 20* *-margin 10 20* *-margin 10 20 0* *-margin 10 20 0 40*	Sets size of margin for each PDF page in points. Default value for each margin: 0. Allowable values: 0-size of page in points; in addition, the sum of the left and right values must be less than the width of the page, and the sum of the top and bottom values must be less than the height of the page. ***-margin left top right bottom*** **-margin 20**: sets the left margin to 20 points. **-margin 10 20**: sets the left margin to 10 points and the top margin to 20 points. **-margin 10 20 0**: sets the left margin to 10 points, the top margin to 20 points, and the right margin to 0 points. **-margin 10 20 0 40**: sets the left margin to 10 points, the top margin to 20 points, the right margin to 0 points, and the bottom margin to 40 points.
-b	*[-b]* *e.g.* *-b*	Uses the filename of the image(s) to create bookmark(s) for the output PDF.
-sp	*[-sp <string>]* *e.g.* *-sp 123* *-sp welcome*	Sets the document open password of the output PDF as the "string". By default, there is no password.
-title	*<-title <string>>* *e.g.* *-title "Foxit PDF Toolkit User Manual"*	Sets title of PDF files.
-subject	*<-subject <string>>* *e.g.* *-subject "Foxit PDF Toolkit"*	Sets subject of PDF files.

Option	Parameter	Description
-keywords	*[-keywords <string>]* *e.g.* *-keywords "Foxit"*	Sets keywords of PDF files.
-author	*[-author <string>]* *e.g.* *-author "Jessie"*	Sets author of PDF files.
-creator	*[-creator <string>]* *e.g.* *-creator "Foxit PhantomPDF"* *-creator "Foxit Reader"* *-creator "Microsoft® Word 2013"*	Sets creator of PDF files. **Note** It indicates the name of the application that created the source document from which a PDF is generated. For example, if the source document was created from Microsoft® *Word 2013, then you can set the creator to "*Microsoft® *Word 2013".*
-r	*[-r [integer]]* *e.g.* *-r* *-r 0* *-r 1* *-r 2* …	Specifies the number of layers to recurse when the input is a folder. • **-r 0 <-r>**: searches the full folders. • **-r 1**: searches only the current folder. • **-r 2**: searches the current folder and its sub-folders … **Note** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input folder will be skipped if it is secured and the messages will be displayed.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2*	Specifies the number of CPU threads to use. The default value is 1.
-log	*[-log <string>]* *e.g. -log d:\a.log*	Writes log information into a logfile at the specified existing path.

Option	Parameter	Description
-l	*[-l <integer>]* *e.g.* *-l 1* *-l 2* *-l 3* *-l 4*	Sets the log level. The default is 4. • **-l 1**: logs messages only concerning program crashes. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1. • **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>* *e.g.* *-register* 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foxit	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.
-help/-h	*[-help/-h]* *e.g.* *-help* *-h*	Prints the usage information.
-version/-v	*[-version/-v]* *e.g.* *-version* *-v*	Prints the version information.
-license	*[-license]* *e.g. -license*	Prints the license agreement.

3.1.3 Basic Usage

3.1.3.1 Input and Output

a) Input (-i)

- The input files should be a single image file, multiple image files, or a folder. Users are not able to input multiple folders, as well as a mixture composed of folders and image files. For example:

-i c:\input\1.jpg	(a single image file)
-i “c:\input\1.jpg” “c:\input\2.tif”	(multiple image files)
-i c:\input	(a single folder)

Note *If the input files are multiple image files, it is recommended to enclose each input path with quotation marks (“ ”). In this manual, we add (“ ”) whenever the input files are multiple image files.*

- It supports relative paths if the input files are in the current working folder. Users can input just the name of the image files or folder, instead of an absolute path. For example:

-i test\3.bmp	(“test\3.bmp” is in the current working folder)
-i test	(“test” folder is in the current working folder)

- It also supports wildcard characters, which are used to process multiple image files in specified formats. For example:

-i “c:\input*.jpg”	(Only convert images with JPG format)
-i “c:\input*.jpg” “c:\input*.tif”	(Only convert images with JPG and TIF formats)
-i “test*.png”	(Only convert images with PNG format)

Note *When using wildcard characters in the input files, it is recommended to enclose the input files with quotation marks (“ ”). In this manual, we add (“ ”) whenever the input files contain wildcard characters.*

b) Output (-o)

- There are two output formats:
  - Combine multiple image files into one PDF file
  - Convert each image file into individual PDF files
- If you want to combine multiple image files into one PDF file, you should specify the output path of a PDF file. If you want to convert each image file into individual PDF files, you should specify the output path of a folder. For example:

-o d:\output\one.pdf	(Combine multiple image files into one PDF file)
-o d:\output	(Convert each image file into individual PDF files)

Note *The specified output path must already exist.*

- The output also supports relative paths if the specified output location is in the current working folder. Users can input just the name of the PDF file or output folder, instead of an absolute path. For example:

  -o output\one.pdf ("output" folder is in the current working folder)
  -o output ("output" folder is in the current working folder)

- When the input is one or more image files, or if it includes wildcard characters, the output should be a single PDF file. For example, the first following line combines the "3.bmp" and "4.gif" image files into "out.pdf" file, where the "test" and "output" folders are both in the current working folder. And the second line combines all the ".jpg" and ".tif" image files in the "c:\input" folder into "out.pdf" in "d:\output" folder.

  fimg2pdf -i "test\3.bmp" "test\4.gif" -o output\out.pdf
  fimg2pdf -i "c:\input*.jpg" "c:\input*.tif" –o d:\output\out.pdf

Usage Examples

1) Convert each image file into individual PDF files:

   fimg2pdf -i test -o output ("test" and "output" folders are both in the current working folder)
   fimg2pdf -i c:\input -o d:\output

2) Combine multiple image files into one PDF file:

   fimg2pdf -i "test\3.bmp" "test\4.gif" -o output\one.pdf
   fimg2pdf -i c:\input\1.jpg -o d:\output\one.pdf
   fimg2pdf -i "c:\input\1.jpg" "c:\input\2.tif" -o d:\output\one.pdf
   fimg2pdf -i c:\input -o d:\output\one.pdf
   fimg2pdf -i "c:\input*.jpg" -o d:\output\one.pdf

3.1.3.2 PDF/Page Settings

a) Page size setting (-width, -height)

- The optional arguments (**-width**) and (**-height**) are used to set the page width and height for the output PDF file in points.

  Note *The –width and –height options must be used together with a set value greater than 0.*

Usage Example

1) Set the page width and height to 400 and 300 for the output PDF file (-width 400 -height 300)

```
fimg2pdf -i "c:\input\1.jpg" "c:\input\2.tif" -o d:\output\one.pdf -width 400 -height 300
fimg2pdf -i c:\input -o d:\output -width 400 -height 300
fimg2pdf -i "c:\input\*.jpg" -o d:\output\one.pdf -width 400 -height 300
```

b) Resolution (DPI) setting (-dpi)

- The optional argument (**-dpi**) is used to set the resolution for the output PDF file. By default, the DPI (Dots Per Inch) is 72, which is the typical resolution for web images. For more details about this argument, please refer to section 3.1.2 "Command Line Summary".

Note

- *If the program failed to get the DPI value of the input image when the (**-dpi**) is set to 1, the DPI value will be 96.*
- *The options -weight, -height and -dpi should not be used together.*
  - If you only use -width and -dpi, or -height and -dpi, the width or height setting will be omitted.
  - If you use -width, -height and -dpi, the dpi setting will be omitted.

Usage Examples

1) Use the default image width and height information (-dpi 0)

```
fimg2pdf -i "c:\input\1.jpg" "c:\input\2.tif" -o d:\output\one.pdf -dpi 0
fimg2pdf -i c:\input –o d:\output -dpi 0
fimg2pdf -i "c:\input\*.jpg" –o d:\output\one.pdf -dpi 0
```

2) Use the DPI information of the original image (-dpi 1)

```
fimg2pdf -i "c:\input\1.jpg" "c:\input\2.tif" -o d:\output\one.pdf -dpi 1
fimg2pdf -i c:\input –o d:\output -dpi 1
fimg2pdf -i "c:\input\*.jpg" –o d:\output\one.pdf -dpi 1
```

3) Set the DPI to 300 for the output PDF file (-dpi 300)

```
fimg2pdf -i "c:\input\1.jpg" "c:\input\2.tif" -o d:\output\one.pdf -dpi 300
```

fimg2pdf -i c:\input –o d:\output -dpi 300
fimg2pdf -i "c:\input*.jpg" –o d:\output\one.pdf -dpi 300

c) Bookmark (-b)

- The optional argument (-b) is used to create bookmarks for the output PDF file. The name of the bookmarks is the name of the images.

Usage Example

1) Create bookmarks for the output PDF file (-b)

fimg2pdf -i "c:\input\1.jpg" "c:\input\2.tif" -o d:\output\one.pdf -b
fimg2pdf -i c:\input -o d:\output\one.pdf -b
fimg2pdf -i "c:\input*.jpg" -o d:\output\one.pdf -b

d) Margin setting (-margin)

- The optional argument (-margin) is used to set size of margin for each PDF page in points. By default, the output PDF page has no margin. For more details about this argument, please refer to section 3.1.2 "Command Line Summary".

Note *The sum of the left and right values must be less than the width of the page, and the sum of the top and bottom values must be less than the height of the page.*

Usage Example

1) Set the left margin to 20 points (-margin 20)

fimg2pdf -i "c:\input\1.jpg" "c:\input\2.tif" -o d:\output\one.pdf -margin 20
fimg2pdf -i c:\input -o d:\output\one.pdf -margin 20
fimg2pdf -i "c:\input*.jpg" -o d:\output\one.pdf -margin 20

2) Set the left margin to 20 points, and the top margin to 10 points (-margin 20 10)

fimg2pdf -i "c:\input\1.jpg" "c:\input\2.tif" -o d:\output\one.pdf -margin 20 10
fimg2pdf -i c:\input -o d:\output\one.pdf -margin 20 10
fimg2pdf -i "c:\input*.jpg" -o d:\output\one.pdf -margin 20 10

3) Set the left margin to 10 points, the top margin to 10 points and the right margin to 30 points (-margin 10 10 30)

fimg2pdf -i “c:\input\1.jpg” “c:\input\2.tif” -o d:\output\one.pdf -margin 10 10 30
fimg2pdf -i c:\input -o d:\output\one.pdf -margin 10 10 30
fimg2pdf -i “c:\input*.jpg” -o d:\output\one.pdf -margin 10 10 30

4) Set the left margin to 10 points, the top margin to 10 points, the right margin to 30 points and the bottom margin to 20 points (-margin 10 10 30 20)

fimg2pdf -i “c:\input\1.jpg” “c:\input\2.tif” -o d:\output\one.pdf -margin 10 10 30 20
fimg2pdf -i c:\input -o d:\output\one.pdf -margin 10 10 30 20
fimg2pdf -i “c:\input*.jpg” -o d:\output\one.pdf -margin 10 10 30 20

e) Password setting (-sp)

- The optional argument (**-sp**) is used to set the open password for the output PDF file. By default, the output PDF file has no open password.

Usage Example

1) Set the open password to “welcome” for the output PDF files (-sp welcome)

fimg2pdf -i “c:\input\1.jpg” “c:\input\2.tif” -o d:\output\one.pdf -sp welcome
fimg2pdf -i c:\input –o d:\output -sp welcome
fimg2pdf -i “c:\input*.jpg” –o d:\output\one.pdf -sp welcome

3.1.3.3 Document Metadata Settings

a) Title (-title)

- The optional argument (**-title**) is used to set title of PDF files.

Usage Example

1) Set document title to “Foxit PDF Toolkit User Manual” (-title “Foxit PDF Toolkit User Manual”)

fimg2pdf -i “c:\input\1.jpg” “c:\input\2.tif” -o d:\output\one.pdf -title “Foxit PDF Toolkit User Manual”
fimg2pdf -i c:\input -o d:\output\one.pdf -title “Foxit PDF Toolkit User Manual”

fimg2pdf -i “c:\input*.jpg” -o d:\output\one.pdf -title “Foxit PDF Toolkit User Manual”

b) Subject (-subject)

- The optional argument (**-subject**) is used to set subject of PDF files.

Usage Example

1) Set document subject to “Foxit PDF Toolkit” (-subject “Foxit PDF Toolkit”)

fimg2pdf -i “c:\input\1.jpg” “c:\input\2.tif” -o d:\output\one.pdf -subject “Foxit PDF Toolkit”
fimg2pdf -i c:\input -o d:\output\one.pdf -subject “Foxit PDF Toolkit”
fimg2pdf -i “c:\input*.jpg” -o d:\output\one.pdf -subject “Foxit PDF Toolkit”

c) Keywords (-keywords)

- The optional argument (**-keywords**) is used to set keywords of PDF files.

Usage Example

1) Set document keywords to “toolkit” (-keywords “toolkit”)

fimg2pdf -i “c:\input\1.jpg” “c:\input\2.tif” -o d:\output\one.pdf -keywords “toolkit”
fimg2pdf -i c:\input -o d:\output\one.pdf -keywords “toolkit”
fimg2pdf -i “c:\input*.jpg” -o d:\output\one.pdf -keywords “toolkit”

d) Author (-author)

- The optional argument (**-author**) is used to set author of PDF files.

Usage Example

1) Set document author to “Jessie” (-author “Jessie”)

fimg2pdf -i “c:\input\1.jpg” “c:\input\2.tif” -o d:\output\one.pdf -author “Jessie”
fimg2pdf -i c:\input -o d:\output\one.pdf -author “Jessie”
fimg2pdf -i “c:\input*.jpg” -o d:\output\one.pdf -author “Jessie”

e) Creator (-creator)

- The optional argument (**-creator**) is used to set file creation application information of PDF files.

Usage Example

1) Set document creator to "Foxit Reader" (-creator "Foxit Reader")

```
fimg2pdf -i "c:\input\1.jpg" "c:\input\2.tif" -o d:\output\one.pdf -creator "Foxit Reader"
fimg2pdf -i c:\input -o d:\output\one.pdf -creator "Foxit Reader"
fimg2pdf -i "c:\input\*.jpg" -o d:\output\one.pdf -creator "Foxit Reader"
```

3.1.3.4 Recursion Depth of Sub-folders

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard character like "c:\input*.jpg". By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.1.2 "Command Line Summary".

Usage Examples

1) Search the full folders (-r or -r 0)

```
fimg2pdf -i test -o output -r 0
fimg2pdf -i c:\input -o d:\output -r 0
fimg2pdf -i "c:\input\*.jpg" -o d:\output\one.pdf -r 0
fimg2pdf -i test -o output -r
fimg2pdf -i c:\input -o d:\output -r
fimg2pdf -i "c:\input\*.jpg" -o d:\output\one.pdf -r
```

2) Search only the current folder (-r 1)

```
fimg2pdf -i test -o output -r 1
fimg2pdf -i c:\input -o d:\output -r 1
fimg2pdf -i "c:\input\*.jpg" -o d:\output\one.pdf -r 1
```

Note *If you don't use this argument, the current folder will be searched by default. For example:*

```
fimg2pdf -i test -o output
fimg2pdf -i c:\input -o d:\output
fimg2pdf -i "c:\input\*.jpg" -o d:\output\one.pdf
```

3) Search the current folder and its sub-folders (-r 2)

fimg2pdf -i test -o output -r 2
fimg2pdf -i c:\input -o d:\output -r 2
fimg2pdf -i "c:\input*.jpg" -o d:\output\one.pdf -r 2

3.1.3.5 Multi-thread Support

- The optional argument (**-t**) indicates the number of threads that are used to speed up batch conversion by making full use of the CPU. By default, the number of the threads is 1.

  Note *It is recommended that you set the value of the number according to your computer's CPU configuration.*

Usage Example

1) Set the number of threads to 3 (-t 3)

   fimg2pdf -i test -o output -t 3
   fimg2pdf -i c:\input -o d:\output -t 3
   fimg2pdf -i "c:\input*.jpg" -o d:\output\one.pdf -t 3

3.1.3.6 Other Optional Arguments

a) Log file (-log<logfile> -l<log level>)

- The optional argument (**-log**) indicates the path of logfile, where the log message is placed. The argument (**-l**) indicates the log level. It is valid only when (**-log**) is used. By default, the log level is 4. For more details about this argument, please refer to section 3.1.2 "Command Line Summary".

Usage Example

1) Save the log file to "d:\output\image2pdf.log" and set the log level to 3 (-log d:\output\image2pdf.log -l 3)

   fimg2pdf -i c:\input -o d:\output -log d:\output\image2pdf.log -l 3

b) Register information (-register <code> <licensee>)

- The optional argument (**-register**) is used to register the command line tool. The **<code>** is the activation code from Foxit and **<licensee>** is the licensee name designated by the users.

Usage Example

1) Register the image2pdf tool with the code "77505-010G0-G1000-XMQ8D-2CR7R-TPBEI" and the licensee "Foxit" (-register 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foixt)

   fimg2pdf -register 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foixt

c) License agreement (-license)

- The optional argument (**-license**) is used to print the license agreement.

Usage Example

1) Print the license information (-license)

   fimg2pdf -license

d) Version information (-version/-v)

- The optional argument (-version/-v) is used to print the version information.

Usage Example

1) Print the version information (-version/-v)

   fimg2pdf -version
   fimg2pdf -v

e) Help information (-help/-h)

- The optional argument (**-help/-h**) is used to print the usage information.

Usage Example

1) Print the usage information (-help/-h)

   fimg2pdf -help
   fimg2pdf -h

3.2 Office2PDF

3.2.1 Basic Syntax

foffice2pdf *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> [-pdfa] [-b <bookmark level>] [-scale] [-op <password>] [-r [recursion]] [-t <threads>] [-log <logfile>] [-l <level>]*

foffice2pdf *-register <code> <licensee>*

foffice2pdf *-license*

foffice2pdf *-version/-v*

foffice2pdf *-help/-h*

Note:

- <> required
- [ ] optional
- / mutually exclusive
- A space is needed between the command line argument and the value

Only the <*-i <srcfile/srcfolder>>* and <*-o <destfile/destfolder>>* arguments are actually required. All others are optional, which are available for controlling the output PDF files as desired. The arguments could be given in any order. Full details on each are explained in the following section.

3.2.2 Command Line Summary

Note For some arguments whose values are strings, users can choose whether to add *quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i c:\input\1.doc* *-i c:\input* *-i "c:\input*.ppt"*	Specifies the input file to be converted. ▪ The input string can be the name of a single Microsoft Office file (.doc, .docx, .xls, .xlsx, .ppt, .pptx) or a folder. ▪ The file name can contain the wildcard character (*). For example, use *.doc to include all Doc files in a given folder. **Note** The wildcard character (*.*) is currently not supported.

Option	Parameter	Description
-o	*<-o <string>>* *e.g.* *-o D:\output\1.pdf* *-o D:\output*	Specifies the path of the output PDF file or folder. ▪ If the input is a single Microsoft Office file, the output should be a single PDF file, (e.g. *-o D:\output\1.pdf*). ▪ If the input is a folder, the output should be a folder, (e.g. *-o D:\output*). **Note** The specified output path must already exist.
-pdfa	*[-pdfa]* *e.g.* *-pdfa*	Specifies that the output PDF file(s) should be compliant with the PDF/A standard.
-b	*[-b <integer>]* *e.g.* *-b 0* *-b 1* *-b 2*	Creates bookmarks for the output PDF file. If not set, there are no PDF bookmarks. • **-b 0**: Not to create PDF bookmarks. • **-b 1**: Create PDF bookmarks using headings of a Microsoft Word file. • **-b 2**: Create PDF bookmarks using bookmarks of a Microsoft Word file. **Note** This argument (**-b**) is valid only for Microsoft Word files.
-scale	*[-scale<integer>]* *e.g.* *-scale 0* *-scale 1* *-scale 2* *-scale 3*	Specifies the conversion mode for Microsoft Excel files. The default is 1. • **-scale 0**: No scaling. Convert sheets at their actual size. • **-scale 1**: Fit all columns on one page. Scale every sheet so that it is one page wide. • **-scale 2**: Fit all rows on one page. Scale every sheet so that it is one page high. • **-scale 3**: Fit sheet on one page. Scale every sheet so that it fits on one page. **Note** This argument (**-scale**) is supported only on versions higher than Microsoft Office 2007 and is valid only for Microsoft Excel files.

Option	Parameter	Description
-op	*[-op<string>]*	Specifies the open password for the input file. Not required if the input file is not password protected. **Note** The output PDF file will not retain the open password from the input file.
-r	*[-r [integer]]* *e.g.* *-r* *-r 0* *-r 1* *-r 2* … **Note** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input Microsoft Office file or folder will be skipped if it is secured and the messages will be displayed.	Specifies the number of layers to recurse when the input is a folder. • **-r 0 <-r>**: searches the full folders. • **-r 1**: searches only the current folder. • **-r 2**: searches the current folder and its sub-folders …
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2*	Specifies the number of CPU threads to use. The default value is 1.
-log	*[-log <string>]* *e.g.* *-log d:\a.log*	Writes log information into a logfile at the specified existing path.
-l	*[-l <integer>]* *e.g.* *-l 1* *-l 2* *-l 3* *-l 4*	Sets the log level. The default is 4. • **-l 1**: logs messages only concerning program crashes. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1.

Option	Parameter	Description
		• **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>* *e.g.* *-register* 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foxit	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.
-help/-h	*[-help/-h]* *e.g.* *-help* *-h*	Prints the usage information.
-version/-v	*[-version/-v]* *e.g.* *-version* *-v*	Prints the version information.
-license	*[-license]* *e.g.* *-license*	Prints the license agreement.

3.2.3 Basic Usage

3.2.3.1 Input and Output

a) Input (-i)

➢ The input file should be a single Microsoft Office file or a folder. Users are not able to input multiple Office files or folders, as well as a mixture composed of folders and Office files. For example:

-i c:\input\1.doc (a single Microsoft Office file)
-i c:\input (a single folder)

- It supports relative paths if the input file is in the current working folder. Users can input just the name of the Microsoft Office file or folder, instead of an absolute path. For example:

  -i test\2.xls ("test\2.xls" is in the current working folder)
  -i test ("test" folder is in the current working folder)

- It also supports wildcard characters, which are used to process multiple office files in specified formats. For example:

  -i "c:\input*.doc" (Only convert Office files with DOC format)
  -i "test*.xls" (Only convert Office files with XLS format)

  Note *When using wildcard characters in the input files, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

b) Output (-o)

- If the input is a single Microsoft Office file, you should specify the output path of a PDF file. If the input is a single folder, you should specify the output path of a folder. For example:

  -o d:\output\ 1.pdf
  -o d:\output

  Note *The specified output path must already exist.*

- The output also supports relative paths if the specified output location is in the current working folder. Users can input just the name of the PDF file or output folder, instead of an absolute path. For example:

  -o output\2.pdf ("output\2.pdf" is in the current working folder)
  -o output ("output" folder is in the current working folder)

Usage Examples

1) Convert Microsoft Office documents into PDF files:

```
foffice2pdf -i test\2.xls -o output\2.pdf
foffice2pdf -i c:\input -o d:\output
foffice2pdf -i "c:\input\*.doc" -o d:\output
foffice2pdf -i test -o output
```

3.2.3.2 Bookmark

- The optional argument (**-b**) is used to create bookmarks for the output PDF file. If not set, there are no PDF bookmarks by default. For more details about this argument, please refer to section 3.2.2 "Command Line Summary".

  Note *This argument is valid only for Microsoft Word files.*

Usage Example

1) Create PDF bookmarks using headings of a Microsoft Word file (-b 1)

```
foffice2pdf -i c:\input\1.doc -o d:\output\1.pdf -b 1
foffice2pdf -i c:\input -o d:\output -b 1
foffice2pdf -i test -o output -b 1
foffice2pdf -i "c:\input\*.doc" -o d:\output -b 1
```

2) Create PDF bookmarks using bookmarks of a Microsoft Word file (-b 2)

```
foffice2pdf -i c:\input\1.doc -o d:\output\1.pdf -b 2
foffice2pdf -i c:\input -o d:\output -b 2
foffice2pdf -i test -o output -b 2
foffice2pdf -i "c:\input\*.doc" -o d:\output -b 2
```

3.2.3.3 Scale

- The optional argument (**-scale**) is used to specify the conversion mode for Microsoft Excel files. For more details about this argument, please refer to section 3.2.2 "Command Line Summary".

  Note *This argument is supported only on versions higher than Microsoft Office 2007 and is valid only for Microsoft Excel files.*

Usage Example

1) Convert sheets at their actual size (-scale 0)

```
foffice2pdf -i c:\input\1.xlsx -o d:\output\1.pdf -scale 0
foffice2pdf -i c:\input -o d:\output -scale 0
foffice2pdf -i test -o output -scale 0
foffice2pdf -i "c:\input\*.xls" -o d:\output -scale 0
```

2) Fit all columns on one page (-scale 1)

foffice2pdf -i c:\input\1.xlsx -o d:\output\1.pdf -scale 1
foffice2pdf -i c:\input -o d:\output -scale 1
foffice2pdf -i test -o output -scale 1
foffice2pdf -i "c:\input*.xls" -o d:\output -scale 1

3) Fit all rows on one page (-scale 2)

foffice2pdf -i c:\input\1.xlsx -o d:\output\1.pdf -scale 2
foffice2pdf -i c:\input -o d:\output -scale 2
foffice2pdf -i test -o output -scale 2
foffice2pdf -i "c:\input*.xls" -o d:\output -scale 2

4) Fit sheet on one page (-scale 3)

foffice2pdf -i c:\input\1.xlsx -o d:\output\1.pdf -scale 3
foffice2pdf -i c:\input -o d:\output -scale 3
foffice2pdf -i test -o output -scale 3
foffice2pdf -i "c:\input*.xls" -o d:\output -scale 3

3.2.3.4 PDF/A Support

- The optional argument (**-pdfa**) is used to specify the output PDF file to be compliant with the PDF/A Standard. PDF/A is an ISO standardized version of the PDF, specialized for the digital preservation of electronic documents defining provisions for long-term archiving.

  Note *It only supports PDF/A-1b Standard. For more details about PDF/A-1b, please refer to PDF Reference.*

Usage Example

1) Specify the output PDF file to be compliant with PDF/A Standard (-pdfa)

foffice2pdf -i c:\input\1.doc -o d:\output\1.pdf -pdfa
foffice2pdf -i c:\input -o d:\output -pdfa
foffice2pdf -i test -o output -pdfa
foffice2pdf -i "c:\input*.doc" -o d:\output -pdfa

3.2.3.5 Open Password

- The optional argument (**-op**) indicates the open password for a password-protected input Microsoft Office file. It is not required if the input file is not password protected.

Note *The output PDF file will not retain the open password from the input file.*

Usage Example

1) Specify the open password for a password-protected input Office file (-op 123)

```
foffice2pdf -i c:\input\1.doc -o d:\output\1.pdf -op 123
foffice2pdf -i test\2.xls -o output\2.pdf -op 123
```

2) Specify the open password for all input Office files that have been protected with the same password (-op 123)

```
foffice2pdf -i c:\input -o d:\output -op 123
foffice2pdf -i test -o output -op 123
foffice2pdf -i "c:\input\*.doc" -o d:\output -op 123
```

Note *It only supports typing one value for the argument (**-op**). Only files with the same open password can be processed together and files with different open password need to be processed separately.*

3.2.3.6 Recursion Depth of Sub-folders

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard character like "c:\input*.doc". By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.2.2 "Command Line Summary".

Usage Examples

1) Search the full folders (-r or -r 0)

```
foffice2pdf -i c:\input -o d:\output -r
foffice2pdf -i test -o output -r
foffice2pdf -i "c:\input\*.doc" -o d:\output -r
foffice2pdf -i c:\input -o d:\output -r 0
foffice2pdf -i test -o output -r 0
foffice2pdf -i "c:\input\*.doc" -o d:\output -r 0
```

2) Search only the current folder (-r 1)

foffice2pdf -i c:\input -o d:\output -r 1
foffice2pdf -i test -o output -r 1
foffice2pdf -i "c:\input*.doc" -o d:\output -r 1

Note *If you don't use this argument, the current folder will be searched by default. For example:*

foffice2pdf -i c:\input -o d:\output
foffice2pdf -i test -o output
foffice2pdf -i "c:\input*.doc" -o d:\output

3) Search the current folder and its sub-folders (-r 2)

foffice2pdf -i c:\input -o d:\output -r 2
foffice2pdf -i test -o output -r 2
foffice2pdf -i "c:\input*.doc" -o d:\output -r 2

3.2.3.7 Multi-thread Support

- The optional argument (**-t**) indicates the number of threads that are used to speed up batch conversion by making full use of the CPU. By default, the number of threads is 1.

  Note *It is recommended that you set the value of the number according to your computer's CPU configuration.*

Usage Example

1) Set the number of threads to 3 (-t 3)

foffice2pdf -i c:\input -o d:\output -t 3
foffice2pdf -i test -o output -t 3
foffice2pdf -i "c:\input*.doc" -o d:\output -t 3

3.2.3.8 Other Optional Arguments

a) Log file (-log<logfile> -l<log level>)

- The optional argument (**-log**) indicates the path of logfile, where the log message is placed. The argument (**-l**) indicates the log level. It is valid only when (**-log**) is used. By default, the log level is 4. For more details about this argument, please refer to section 3.2.2 "Command Line Summary".

Usage Example

1) Save the log file to "d:\output\office2pdf.log" and set the log level to 3 (-log d:\output\office2pdf.log -l 3)

```
foffice2pdf -i c:\input\1.doc -o d:\output\1.pdf -log d:\output\office2pdf.log -l 3
```

b) Register information (-register <code> <licensee>)

- The optional argument (**-register**) is used to register the command line tool. The **<code>** is the activation code from Foxit and **<licensee>** is the licensee name designated by the users.

Usage Example

2) Register the office2pdf tool with the code "77505-010G0-G1000-XMQ8D-2CR7R-TPBEI" and the licensee "Foxit" (-register 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foixt)

```
foffice2pdf -register 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foixt
```

c) License agreement (-license)

- The optional argument (**-license**) is used to print the license agreement.

Usage Example

1) Print the license information (-license)

```
foffice2pdf -license
```

d) Version information (-version/-v)

- The optional argument (-version/-v) is used to print the version information.

Usage Example

1) Print the version information (-version/-v)

```
foffice2pdf -version
foffice2pdf -v
```

e) Help information (-help/-h)

- The optional argument (**-help/-h**) is used to print the usage information.

Usage Example

1) Print the usage information (-help/-h)

```
foffice2pdf -help
foffice2pdf -h
```

3.3 PDFWatermark

3.3.1 Basic Syntax

fpdfwm *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> <-conf <xmlfile>> [-op <password>]*
[-title <title>] [-subject <subject>] [-keywords <keywords>] [-author <author>] [-creator <creator>]
[-r [recursion]] [-t <threads>] [-log <logfile>] [-l <level>]

fpdfwm *-register <code> <licensee>*

fpdfwm *-license*

fpdfwm *-version/-v*

fpdfwm *-help/-h*

Note:

- <> required
- [ ] optional
- / mutually exclusive
- A space is needed between the command line argument and the value

Only the <*-i <srcfile/srcfolder>>*, <*-o <destfile/destfolder>>* and <*-conf <xmlfile>>* arguments are actually required. All others are optional, which are available for controlling the process as desired. The arguments could be given in any order. The XML file is a configuration file generated by the build-in Foxit Configuration Tool. Full details on each are explained in the following section.

3.3.2 Command Line Summary

Note For some arguments whose values are strings, users can choose whether to add *quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i c:\input\1.pdf* *-i c:\input*	Specifies the input file to be processed. ▪ The input string can be the name of a single PDF file or a folder. ▪ The file name can contain the wildcard character (*). For example, use *.pdf to include all PDF files in a given folder. **Note** Wildcard character (*.*) is currently not supported.

Option	Parameter	Description
-o	*<-o <string>>* *e.g.* *-o d:\output\1_wm.pdf* *-o d:\output*	Specifies the path of the output PDF file or folder. ▪ If the input is a PDF file, the output should be a single PDF file, (e.g. *-o d:\output\1_wm.pdf*). ▪ If the input is a folder, the output should be a folder, (e.g. *-o d:\output*). **Note** The specified output path must already exist.
-conf	*<-conf <xmlfile>>* *e.g.* *-conf c:\watermark.xml*	Specifies the configuration file on the PDFWatermark tool. This file should be generated by the build-in Foxit Configuration Tool.
-op	*[-op<string>]*	Specifies the open password for the input file. Not required if the input file is not password protected. **Note** The output PDF file will retain the open password from the input file.
-title	*<-title <string>>* *e.g.* *-title "Foxit PDF Toolkit User Manual"*	Sets title of PDF files.
-subject	*<-subject <string>>* *e.g.* *-subject "Foxit PDF Toolkit"*	Sets subject of PDF files.
-keywords	*[-keywords <string>]* *e.g.* *-keywords "Foxit"*	Sets keywords of PDF files.
-author	*[-author <string>]* *e.g.* *-author "Jessie"*	Sets author of PDF files.

Option	Parameter	Description
-creator	*[-creator <string>]* *e.g.* *-creator "Foxit PhantomPDF"* *-creator "Foxit Reader"* *-creator "Microsoft® Word 2013"*	Sets creator of PDF files. **Note** It indicates the name of the application that created the source document from which a PDF is generated. For example, if the source document was created from Microsoft® *Word 2013, then you can set the creator to "*Microsoft® *Word 2013".*
-r	*[-r [integer]]* *e.g.* *-r* *-r 0* *-r 1* *-r 2* *…*	Specifies the number of layers to recurse when the input is a folder. • **-r 0 <-r>**: searches the full folders. • **-r 1**: searches only the current folder. • **-r 2**: searches the current folder and its sub-folders … **Note** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input PDF file or folder will be skipped if it is secured and the messages will be displayed.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2*	Specifies the number of CPU threads to use. The default value is 1.
-log	*[-log <string>]* *e.g. -log d:\a.log*	Writes log information into a logfile at the specified existing path.
-l	*[-l <integer>]* *e.g.* *-l 1* *-l 2* *-l 3* *-l 4*	Sets the log level. The default is 4. • **-l 1**: logs messages only concerning program crashes. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1.

Option	Parameter	Description
		• **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>* *e.g.* *-register* 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foxit	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.
-help/-h	*[-help/-h]* *e.g.* *-help* *-h*	Prints the usage information.
-version/-v	*[-version/-v]* *e.g.* *-version* *-v*	Prints the version information.
-license	*[-license]* *e.g.* *-license*	Prints the license agreement.

3.3.3 Basic Usage

3.3.3.1 Required Arguments

a) Input (-i)

➢ The input file should be a single PDF file or a folder. Users are not able to input multiple PDF files or folders, as well as a mixture composed of folders and PDF files. For example:

-i c:\input\1.pdf (a single PDF file)
-i c:\input (a single folder)

- It supports relative paths if the input file is in the current working folder. Users can input just the name of the PDF file or folder, instead of an absolute path. For example:

  -i test\2.pdf ("test\2.pdf" is in the current working folder)
  -i test ("test" folder is in the current working folder)

- It also supports wildcard characters, which are used to process multiple PDF files. For example:

  -i "c:\input*.pdf" (Only convert PDF files under "c:\input" folder)
  -i "test*.pdf" (Only convert PDF files under "test" folder)

  Note *When using wildcard characters in the input files, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

b) Output (-o)

- If the input is a single PDF file, you should specify the output path of a PDF file. If the input is a single folder, you should specify the output path of a folder. For example:

  -o d:\output\ 1_wm.pdf (a single PDF file)
  -o d:\output (a single folder)

  Note *The specified output path must already exist.*

- The output supports relative path if the specified output location is in the current working folder. Users can input just the name of the PDF file or output folder, instead of an absolute path. For example:

  -o output\2_wm.pdf ("output" folder is in the current working folder)
  -o output ("output" folder is in the current working folder)

c) XML Configuration File (-conf)

- The XML configuration file argument (**-conf**) is required in the command line. The configuration file contains the setting information of watermark, which is generated by the build-in Foxit Configuration Tool (*fpdfwmconf.exe* or *fpdfwmconf64.exe*). For more details about watermark settings, please refer to section 4.1.1 "Watermark Settings". Users should input the path of an XML file. For example:

  -conf c:\conf_wm.xml

- It also supports relative paths if the specified XML configuration file is in the current working folder. Users can input just the name of the XML file, instead of an absolute path. For example:

  -conf conf_wm.xml (conf_wm.xml is in the current working folder)

Usage Examples

1) Add a watermark into PDF files:

   fpdfwm -i test -o output -conf conf_wm.xml
   fpdfwm -i "c:\input*.pdf" -o d:\output -conf c:\conf_wm.xml
   fpdfwm -i c:\input\1.pdf -o d:\output\1_wm.pdf -conf c:\conf_wm.xml

3.3.3.2 Open Password

- The optional argument (**-op**) indicates the open password for a password-protected input PDF file. It is not required if the input file is not password protected.

  Note *The output PDF file will retain the open password from the input file.*

Usage Example

1) Specify the open password for a password-protected input PDF file (-op 123)

   fpdfwm -i test\2.pdf -o output\2_wm.pdf -conf c:\conf_wm.xml -op 123
   fpdfwm -i c:\input\1.pdf -o d:\output\1_wm.pdf -conf c:\conf_wm.xml -op 123

2) Specify the open password for all input PDF files that have been protected with the same password (-op 123)

   fpdfwm -i test -o output -conf conf_wm.xlm -op 123
   fpdfwm -i c:\input -o d:\output -conf c:\conf_wm.xml -op 123

   Note *It only supports typing one value for the argument (**-op**). Only files with the same open password can be processed together and files with different open password need to be processed separately.*

3.3.3.3 Document Metadata Settings

a) Title (-title)

- The optional argument (**-title**) is used to set title of PDF files.

Usage Example

1) Set document title to "Foxit PDF Toolkit User Manual" (-title "Foxit PDF Toolkit User Manual")

```
fpdfwm -i test -o output -conf conf_wm.xml -title "Foxit PDF Toolkit User Manual"
fpdfwm -i "c:\input\*.pdf" -o d:\output -conf c:\conf_wm.xml -title "Foxit PDF Toolkit User Manual"
fpdfwm -i c:\input\1.pdf -o d:\output\1_wm.pdf -conf c:\conf_wm.xml -title "Foxit PDF Toolkit User Manual"
```

b) Subject (-subject)

- The optional argument (**-subject**) is used to set subject of PDF files.

Usage Example

1) Set document subject to "Foxit PDF Toolkit" (-subject "Foxit PDF Toolkit")

```
fpdfwm -i test -o output -conf conf_wm.xml -subject "Foxit PDF Toolkit"
fpdfwm -i "c:\input\*.pdf" -o d:\output -conf c:\conf_wm.xml -subject "Foxit PDF Toolkit"
fpdfwm -i c:\input\1.pdf -o d:\output\1_wm.pdf -conf c:\conf_wm.xml -subject "Foxit PDF Toolkit"
```

c) Keywords (-keywords)

- The optional argument (**-keywords**) is used to set keywords of PDF files.

Usage Example

1) Set document keywords to "toolkit" (-keywords "toolkit")

```
fpdfwm -i test -o output -conf conf_wm.xml -keywords "toolkit"
fpdfwm -i "c:\input\*.pdf" -o d:\output -conf c:\conf_wm.xml -keywords "toolkit"
fpdfwm -i c:\input\1.pdf -o d:\output\1_wm.pdf -conf c:\conf_wm.xml -keywords "toolkit"
```

d) Author (-author)

- The optional argument (**-author**) is used to set author of PDF files.

Usage Example

1) Set document author to “Jessie” (-author “Jessie”)

fpdfwm -i test -o output -conf conf_wm.xml -author “Jessie”
fpdfwm -i “c:\input*.pdf” -o d:\output -conf c:\conf_wm.xml -author “Jessie”
fpdfwm -i c:\input\1.pdf -o d:\output\1_wm.pdf -conf c:\conf_wm.xml -author “Jessie”

e) Creator (-creator)

- The optional argument (**-creator**) is used to set file creation application information of PDF files.

Usage Example

1) Set document creator to “Foxit Reader” (-creator “Foxit Reader”)

fpdfwm -i test -o output -conf conf_wm.xml -creator “Foxit Reader”
fpdfwm -i “c:\input*.pdf” -o d:\output -conf c:\conf_wm.xml -creator “Foxit Reader”
fpdfwm -i c:\input\1.pdf -o d:\output\1_wm.pdf -conf c:\conf_wm.xml -creator “Foxit Reader”

3.3.3.4 Recursion Depth of Sub-folders

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard character like “c:\input*.pdf”. By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.3.2 “Command Line Summary”.

Usage Examples

1) Search the full folders (-r or -r 0)

fpdfwm -i test -o output -conf conf_wm.xml -r
fpdfwm -i c:\input -o d:\output -conf c:\conf_wm.xml -r
fpdfwm -i test -o output -conf conf_wm.xml -r 0
fpdfwm -i c:\input -o d:\output -conf c:\conf_wm.xml -r 0

2) Search only the current folder (-r 1)

fpdfwm -i test -o output -conf conf_wm.xml -r 1
fpdfwm -i c:\input -o d:\output -conf c:\conf_wm.xml -r 1

Note *If you don’t use this argument, the current folder will be searched by default. For example:*

fpdfwm -i test -o output -conf conf_wm.xml
fpdfwm -i c:\input -o d:\output -conf c:\conf_wm.xml

3) Search the current folder and its sub-folders (-r 2)

fpdfwm -i test -o output -conf conf_wm.xml -r 2
fpdfwm -i c:\input -o d:\output -conf c:\conf_wm.xml -r 2

3.3.3.5 Multi-thread Support

- The optional argument (**-t**) indicates the number of threads that are used to speed up batch programming by making full use of the CPU. By default, the number of the threads is 1.

  Note *It is recommended that you set the value of the number according to your computer's CPU configuration.*

Usage Example

1) Set the number of threads to 3 (-t 3)

fpdfwm -i test -o output -conf conf_wm.xml -t 3
fpdfwm -i c:\input -o d:\output -conf c:\conf_wm.xml -t 3

3.3.3.6 Other Optional Arguments

a) Log file (-log<logfile> -l<log level>)

- The optional argument (**-log**) indicates the path of logfile, where the log message is placed. The argument (**-l**) indicates the log level. It is valid only when (**-log**) is used. By default, the log level is 4. For more details about this argument, please refer to section 3.3.2 "Command Line Summary".

Usage Example

1) Save the log file to "d:\output\watermark.log" and set the log level to 3 (-log d:\output\watermark.log -l 3)

fpdfwm -i c:\input -o d:\output -conf c:\conf_wm.xml -log d:\output\watermark.log -l 3

b) Register information (-register <code> <licensee>)

- The optional argument (**-register**) is used to register the command line tool. The **<code>** is the activation code from Foxit and **<licensee>** is the licensee name designated by the users.

Usage Example

3) Register the pdfwatermark tool with the code "77505-010G0-G1000-XMQ8D-2CR7R-TPBEI" and the licensee "Foxit" (-register 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foixt)

```
fpdfwm -register 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foixt
```

c) License agreement (-license)

- The optional argument (**-license**) is used to print the license agreement.

Usage Example

1) Print the license information (-license)

```
fpdfwm -license
```

d) Version information (-version/-v)

- The optional argument (-version/-v) is used to print the version information.

Usage Example

1) Print the version information (-version/-v)

```
fpdfwm -version
fpdfwm -v
```

e) Help information (-help/-h)

- The optional argument (**-help/-h**) is used to print the usage information.

Usage Example

1) Print the usage information (-help/-h)

```
fpdfwm -help
fpdfwm -h
```

3.4 PDFHeaderFooter

3.4.1 Basic Syntax

fpdfhf *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> <-mode <operation mode>> [-conf <xmlfile>]*
[-overlay] [-op <password>] [-title <title>] [-subject <subject>] [-keywords <keywords>]
[-author <author>] [-creator <creator>] [-r [recursion]] [-t <threads>] [-log <logfile>] [-l <level>]
fpdfhf *-register <code> <licensee>*
fpdfhf *-license*
fpdfhf *-version/-v*
fpdfhf *-help/-h*

Note:

- <> required
- [ ] optional
- / mutually exclusive
- A space is needed between the command line argument and the value

Only the *<-i <srcfile/srcfolder>>*, *<-o <destfile/destfolder>>* and *<-mode <operation mode>>* arguments are actually required. The argument *[-conf <xmlfile>]* is required only when "-mode" is set to 1 or 2. All others are optional, which are available for controlling the process as desired. The arguments could be given in any order. The XML file is a configuration file generated by the build-in Foxit Configuration Tool. Full details on each are explained in the following section.

3.4.2 Command Line Summary

Note For some arguments whose values are strings, users can choose whether to add *quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i c:\input\1.pdf* *-i c:\input*	Specifies the input file to be processed. ▪ The input string can be the name of a single PDF file or a folder. ▪ The file name can contain the wildcard character (*). For example, use *.pdf to include all PDF files in a given folder. **Note** Wildcard character (*.*) is currently not supported.

Option	Parameter	Description
-o	*<-o <string>>* *e.g.* *-o d:\ output\1_hf.pdf* *-o d:\output*	Specifies the path of the output PDF file or folder. ▪ If the input is a PDF file, the output should be a single PDF file, (e.g. *-o d:\output\1_hf.pdf*). ▪ If the input is a folder, the output should be a folder, (e.g. *-o d:\output*). **Note** The specified output path must already exist.
-mode	*<-mode <integer>>* *e.g.* *-mode 1* *-mode 2* *-mode 3*	Specifies the mode to be used. • **-mode 1**: adds a new header/footer. An existing header/footer will be overlaid if the (**-overlay**) argument is set; otherwise, the document will not be modified if a header/footer already exists. • **-mode 2**: replaces an existing header/footer. If none exists in the document, a new header/footer will be added. • **-mode 3**: removes an existing header/footer. If none exists in the document, the document will not be modified.
-conf	*<-conf <xmlfile>>* *e.g.* *-conf c:\input\hf.xml*	Specifies the configuration file on the PDFHeaderFooter tool. This file should be generated by the build-in Foxit Configuration Tool. **Note** This argument (**-conf**) will only be used if (**-mode**) is set to 1 or 2.
-overlay	*[-overlay]* *e.g.* *-overlay*	Overlays an existing header/footer. **Note** This argument (**-overlay**) is valid only when (**-mode**) is set to 1.
-op	*[-op <string>]* *e.g.* *-op 123* *-op welcome*	Specifies the open password for the input file. Not required if the input file is not password protected. **Note** The output PDF file will retain the open password from the input file.

Option	Parameter	Description
-title	*<-title <string>>* *e.g.* *-title "Foxit PDF Toolkit User Manual"*	Sets title of PDF files.
-subject	*<-subject <string>>* *e.g.* *-subject "Foxit PDF Toolkit"*	Sets subject of PDF files.
-keywords	*[-keywords <string>]* *e.g.* *-keywords "Foxit"*	Sets keywords of PDF files.
-author	*[-author <string>]* *e.g.* *-author "Jessie"*	Sets author of PDF files.
-creator	*[-creator <string>]* *e.g.* *-creator "Foxit PhantomPDF"* *-creator "Foxit Reader"* *-creator "Microsoft® Word 2013"*	Sets creator of PDF files. **Note** It indicates the name of the application that created the source document from which a PDF is generated. For example, if the source document was created from Microsoft® *Word 2013, then you can set the creator to "*Microsoft® *Word 2013".*
-r	*[-r [integer]]* *e.g.* *-r* *-r 0* *-r 1* *-r 2* ...	Specifies the number of layers to recurse when the input is a folder. • **-r 0 <-r>**: searches the full folders. • **-r 1**: searches only the current folder. • **-r 2**: searches the current folder and its sub-folders ... **Note** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input PDF file or folder will be skipped if it is secured and the messages will be displayed.

Option	Parameter	Description
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2*	Specifies the number of CPU threads to use. The default value is 1.
-log	*[-log <string>]* *e.g. -log d:\a.log*	Writes log information into a logfile at the specified existing path.
-l	*[-l <integer>]* *e.g.* *-l 1* *-l 2* *-l 3* *-l 4*	Sets the log level. The default is 4. • **-l 1**: logs messages only concerning program crashes. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1. • **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>* *e.g.* *-register* 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foxit	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.
-help/-h	*[-help/-h]* *e.g.* *-help* *-h*	Prints the usage information.
-version/-v	*[-version/-v]* *e.g.* *-version* *-v*	Prints the version information.
-license	*[-license]* *e.g.* *-license*	Prints the license agreement.

3.4.3 Basic Usage

3.4.3.1 Required Arguments

a) Input (-i)

- The input file should be a single PDF file or a folder. Users are not able to input multiple PDF files or folders, as well as a mixture composed of folders and PDF files. For example:

  -i c:\input\1.pdf (a single PDF file)
  -i c:\input (a single folder)

- It supports relative paths if the input file is in the current working folder. Users can input just the name of the PDF file or folder, instead of an absolute path. For example:

  -i test\2.pdf ("test\2.pdf" is in the current working folder)
  -i test ("test" folder is in the current working folder)

- It also supports wildcard characters, which are used to process multiple PDF files. For example:

  -i "c:\input*.pdf" (Only convert PDF files under "c:\input" folder)
  -i "test*.pdf" (Only convert PDF files under "test" folder)

  Note *When using wildcard characters in the input files, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

b) Output (-o)

- If the input is a single PDF file, you should specify the output path of a PDF file. If the input is a single folder, you should specify the output path of a folder. For example:

  -o d:\output\1_hf.pdf (a single PDF file)
  -o d:\output (a single folder)

  Note *The specified output path must already exist.*

- The output supports relative paths if the specified output location is in the current working folder. Users can input just the name of the PDF file or output folder, instead of an absolute path. For example:

-o output\1_hf.pdf ("output" folder is in the current working folder)
-o output ("output" folder is in the current working folder)

c) Operation Mode (-mode)

- The operation mode argument (-mode) is required in the command line, which is used to specify the mode to process header/footer. For more details about this argument, please refer to section 3.4.2 "Command Line Summary".

d) XML Configuration File (-conf)

- The XML configuration file argument (**-conf**) is required in the command line when the argument (**-mode**) is set to 1 or 2. The configuration file contains the setting information of header/footer, which is generated by the build-in Foxit Configuration Tool (*fpdfhfconf.exe* or *fpdfhfconf64.exe*). For more details about header/footer settings, please refer to section 4.2.1 "Header/Footer Settings". Users should input a path of an XML file. For example:

  -conf c:\conf_hf.xml

- It also supports relative paths if the specified XML configuration file is in the current working folder. Users can input just the name of the XML file, instead of an absolute path. For example:

  -conf conf_hf.xml (conf_hf.xml is in the current working folder)

Usage Examples

1) Add a new header/footer into PDF files (-mode 1 -conf conf_hf.xml)

   fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 1 -conf c:\conf_hf.xml
   fpdfhf -i "c:\input*.pdf" -o d:\output -mode 1 -conf c:\conf_hf.xml
   fpdfhf -i test -o output -mode 1 -conf conf_hf.xml

2) Replace the header/footer in PDF files (-mode 2 -conf conf_hf.xml)

   fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 2 -conf c:\conf_hf.xml
   fpdfhf -i "c:\input*.pdf" -o d:\output -mode 2 -conf c:\conf_hf.xml
   fpdfhf -i test -o output -mode 2 -conf conf_hf.xml

3) Remove the header/footer in PDF files (-mode 3)

fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 3
fpdfhf -i "c:\input*.pdf" -o d:\output -mode 3
fpdfhf -i test -o output -mode 3

3.4.3.2 Overlay

- The optional argument (**-overlay**) is used to overlay an existing header/footer in PDF file. It is valid only when the argument (**-mode**) is set to 1 and the PDF file already contains a header/footer.

Usage Example

1) Overlay an existing header/footer in PDF file. (-overlay)

fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 1 -conf c:\conf_hf.xml -overlay
fpdfhf -i c:\input -o d:\output -mode 1 -conf c:\conf_hf.xml -overlay
fpdfhf -i test -o output -mode 1 -conf conf_hf.xml -overlay

3.4.3.3 Open Password

- The optional argument (**-op**) indicates the open password for a password-protected input PDF file. It is not required if the input file is not password protected.

  Note *The output PDF file will retain the open password from the input file.*

Usage Example

1) Specify the open password for a password-protected input PDF file (-op 123)

fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 1 -conf c:\conf_hf.xml -op 123
fpdfhf -i test\2.pdf -o output\2_hf.pdf -mode 2 -conf conf_hf.xml -op 123
fpdfhf -i test\3.pdf -o output\3_hf.pdf -mode 3 -op 123

2) Specify the open password for all input PDF files that have been protected with the same password (-op 123)

fpdfhf -i c:\input -o d:\output -mode 1 -conf c:\conf_hf.xml -op 123
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -op 123
fpdfhf -i test -o output -mode 3 -op 123

Note *It only supports typing one value for the argument (**-op**). Only files with the same open password can be processed together and files with different open password need to be processed separately.*

3.4.3.4 Document Metadata Settings

a) Title (-title)

- The optional argument (**-title**) is used to set title of PDF files.

Usage Example

1) Set document title to "Foxit PDF Toolkit User Manual" (-title "Foxit PDF Toolkit User Manual")

```
fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 3 -title "Foxit PDF Toolkit User Manual"
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -title "Foxit PDF Toolkit User Manual"
fpdfhf -i "c:\input\*.pdf" -o d:\output -mode 1 -conf c:\conf_hf.xml -title "Foxit PDF Toolkit User Manual"
```

b) Subject (-subject)

- The optional argument (**-subject**) is used to set subject of PDF files.

Usage Example

1) Set document subject to "Foxit PDF Toolkit" (-subject "Foxit PDF Toolkit")

```
fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 3 -subject "Foxit PDF Toolkit"
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -subject "Foxit PDF Toolkit"
fpdfhf -i "c:\input\*.pdf" -o d:\output -mode 1 -conf c:\conf_hf.xml -subject "Foxit PDF Toolkit"
```

c) Keywords (-keywords)

- The optional argument (**-keywords**) is used to set keywords of PDF files.

Usage Example

1) Set document keywords to "toolkit" (-keywords "toolkit")

```
fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 3 -keywords "toolkit"
```

fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -keywords “toolkit”
fpdfhf -i “c:\input*.pdf” -o d:\output -mode 1 -conf c:\conf_hf.xml -keywords “toolkit”

d) Author (-author)

- The optional argument (**-author**) is used to set author of PDF files.

Usage Example

1) Set document author to “Jessie” (-author “Jessie”)

fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 3 -author “Jessie”
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -author “Jessie”
fpdfhf -i “c:\input*.pdf” -o d:\output -mode 1 -conf c:\conf_hf.xml -author “Jessie”

e) Creator (-creator)

- The optional argument (**-creator**) is used to set file creation application information of PDF files.

Usage Example

1) Set document creator to “Foxit Reader” (-creator “Foxit Reader”)

fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 3 -creator “Foxit Reader”
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -creator “Foxit Reader”
fpdfhf -i “c:\input*.pdf” -o d:\output -mode 1 -conf c:\conf_hf.xml -creator “Foxit Reader”

3.4.3.5 Recursion Depth of Sub-folders

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard character like “c:\input*.pdf”. By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.4.2 “Command Line Summary”.

Usage Examples

1) Search the full folders (-r or -r 0)

fpdfhf -i c:\input -o d:\output -mode 1 -conf c:\conf_hf.xml -r
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -r

fpdfhf -i test -o output -mode 3 -r
fpdfhf -i c:\input -o d:\output -mode 1 -conf c:\conf_hf.xml -r 0
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -r 0
fpdfhf -i test -o output -mode 3 -r 0

2) Search only the current folder (-r 1)

fpdfhf -i c:\input -o d:\output -mode 1 -conf c:\conf_hf.xml -r 1
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -r 1
fpdfhf -i test -o output -mode 3 -r 1

Note *If you don't use this argument, the current folder will be searched by default. For example:*

fpdfhf -i c:\input -o d:\output -mode 1 -conf c:\conf_hf.xml
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml
fpdfhf -i test -o output -mode 3

3) Search the current folder and its sub-folders (-r 2)

fpdfhf -i c:\input -o d:\output -mode 1 -conf c:\conf_hf.xml -r 2
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -r 2
fpdfhf -i test -o output -mode 3 -r 2

3.4.3.6 Multi-thread Support

- The optional argument (**-t**) indicates the number of threads that are used to speed up batch programming by making full use of the CPU. By default, the number of the threads is 1.

  Note *It is recommended that you set the value of the number according to your computer's CPU configuration.*

Usage Example

1) Set the number of threads to 3 (-t 3)

fpdfhf -i c:\input -o d:\output -mode 1 -conf c:\conf_hf.xml -t 3
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -t 3
fpdfhf -i test -o output -mode 3 -t 3

3.4.3.7 Other Optional Arguments

a) Log file (-log<logfile> -l<log level>)

- The optional argument (**-log**) indicates the path of logfile, where the log message is placed. The argument (**-l**) indicates the log level. It is valid only when (**-log**) is used. By default, the log level is 4. For more details about this argument, please refer to section 3.4.2 "Command Line Summary".

Usage Example

1) Save the log file to "d:\output\hf.log" and set the log level to 3 (-log d:\output\hf.log -l 3)

```
fpdfhf -i c:\input -o d:\output -mode 1 -conf c:\conf_hf.xml -log d:\output\hf.log -l 3
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -log d:\output\hf.log -l 3
fpdfhf -i test -o output -mode 3 -log d:\output\hf.log -l 3
```

b) Register information (-register <code> <licensee>)

- The optional argument (**-register**) is used to register the command line tool. The **<code>** is the activation code from Foxit and **<licensee>** is the licensee name designated by the users.

Usage Example

4) Register the pdfheaderfooter tool with the code "77505-010G0-G1000-XMQ8D-2CR7R-TPBEI" and the licensee "Foxit" (-register 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foixt)

```
fpdfhf -register 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foixt
```

c) License agreement (-license)

- The optional argument (**-license**) is used to print the license agreement.

Usage Example

1) Print the license information (-license)

```
fpdfhf -license
```

d) Version information (-version/-v)

- The optional argument (-version/-v) is used to print the version information.

Usage Example

1) Print the version information (-version/-v)

```
fpdfhf -version
fpdfhf -v
```

e) Help information (-help/-h)

- The optional argument (**-help/-h**) is used to print the usage information.

Usage Example

1) Print the usage information (-help/-h)

```
fpdfhf -help
fpdfhf -h
```

3.5 PDFOptimizer

3.5.1 Basic Syntax

fpdfopt *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> [-dc <algorithm> [DPIAbove] [DPISet]]*
[-cc <algorithm> <level> [blocksize]] [-dm <algorithm> [DPIAbove] [DPISet]] [-cm <algorithm> [level]]
[-rd] [-u] [-d <intlist>] [-cl <intlist>] [-op <password>]
[-r [recursion]] [-t <thread>] [-log <logfile>] [-l <level>]

fpdfopt *-register <code> <licensee>*

fpdfopt *-license*

fpdfopt *-version/-v*

fpdfopt *-help/-h*

Note:

- <> required
- [ ] optional
- / mutually exclusive
- A space is needed between the command line argument and the value

Only the <*-i <srcfile/srcfolder>*> and <*-o <destfile/destfolder>*> arguments are actually required. All others are optional, which are available for controlling the output PDF files as desired. The arguments could be given in any order. Full details on each are explained in the following section.

3.5.2 Command Line Summary

Note For some arguments whose values are strings, users can choose whether to add *quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i C:\input\1.pdf* *-i c:\input*	Specifies the input file to be optimized. ▪ The input string can be the name of a single PDF file or a folder. ▪ The file name can contain the wildcard character (*). For example, use *.pdf to include all PDF files in a given folder. **Note** Wildcard character (*.*) is currently not supported.

Option	Parameter	Description
-o	*<-o <string>>* *e.g.* *-o D:\output\1_opt.pdf* *-o D:\output*	Specifies the path of the output PDF file or folder. ▪ If the input is a PDF file, the output should be a single PDF file, (e.g. *-o D:\output\1_opt.pdf*). ▪ If the input is a folder, the output should be a ▪ folder, (e.g. *-o D:\output*). **Note** The specified output path must already exist.
-dc	*[-dc <string> [integer] [integer]]* *-dc <algorithm> [DPIAbove] [DPISet]* *e.g.* *-dc a 225 150* *-dc s 225 150* *-dc b 225 150* *-dc a* *-dc s* *-dc b*	Downsamples color/grayscale images. ▪ **<algorithm> :** Chooses downsampling algorithms **a**: Average Downsampling **s**: SubSampling **b**: Bicubic Downsampling ▪ [**DPIAbove**]: DPI threshold value. Images with a DPI higher than this value will be downsampled. Default value: 225. Allowable range: DPISet-DPISet*10 ▪ [**DPISet**]: Target DPI value images will be downsampled to if their DPI is above the threshold. Default value: 150. Allowable range: 9-2400. **Note** ▪ The **DPIAbove** and **DPISet** must be set at the same time. ▪ If the argument (**-dc**) is not set, images will not be downsampled.
-cc	*[-cc <string> <string> [integer]]* *-cc <algorithm> <level> [blocksize]* *e.g.* *-cc j min* *-cc j low* *-cc j medium* *-cc j high* *-cc j max*	Compresses color/grayscale images. ▪ <**algorithm**>: Chooses compression algorithm. **j**: JPEG **j2**: JPEG2000 ▪ <**level**>: Chooses quality level. Allowable levels are min, low, medium, high and max. ▪ [**blocksize**]: Chooses block size for JPEG2000 algorithm only. Default value: 256. Allowable values: 128-2048.

Option	Parameter	Description
	-cc j2 min 256 *-cc j2 low 256* *-cc j2 medium 256* *-cc j2 high 256* *-cc j2 max 256*	**Note** If the argument (**-cc**) is not set, images will not be compressed.
-dm	*[-dm <string> [integer] [integer]]* *-dm <algorithm> [DPIAbove] [DPISet]* *e.g.* *-dm a 450 300* *-dm s 450 300* *-dm b 450 300* *-dm a* *-dm s* *-dm b*	Downsamples monochrome images. ▪ **<algorithm> :** Chooses downsampling algorithms **a**: Average Downsampling **s**: SubSampling **b**: Bicubic Downsampling ▪ [**DPIAbove**]: DPI threshold value. Images with a DPI higher than this value will be downsampled. Default value: 450. Allowable range: DPISet-DPISet*10 ▪ [**DPISet**]: Target DPI value images will be downsampled to if their DPI is above the threshold. Default value: 300. Allowable range: 9-2400. **Note** ▪ The **DPIAbove** and **DPISet** must be set at the same time. ▪ If the argument (**-dm**) is not set, images will not be downsampled.
-cm	*[-cm <string> [string]]* *-cm <algorithm> [level]* *e.g.* *-cm jbig2 lossless* *-cm jbig2 lossy* *-cm ccitt* *-cm runlength*	Compresses monochrome images. ▪ <**algorithm**>: Chooses compression algorithm. **jbig2**: JBIG2 **ccitt**: CCITT Group 4 **runlength**: RUN LENGTH ▪ [**level**]: Chooses quality level for JBIG2 algorithm only. Allowable levels are lossless and lossy. **Note** If the argument (**-cm**) is not set, images will not be compressed.
-rd	*[-rd]* *e.g. -rd*	If this flag is set, images will be optimized only if there is a reduction in size.

Option	Parameter	Description
-u	*[-u]* *e.g.* *-u*	Unembeds all fonts in selected PDF document(s).
-d	*[-d <string>]* *-d <intlist>* *e.g.* *-d "1"* *-d "1,3,11"* *-d "2-11"*	Discards objects and user data: • 1: discards all form submission, import and reset actions. • 2: flattens form fields. • 3: discards all JavaScript actions. • 4: discards embedded page thumbnails. • 5: discards embedded print settings. • 6: discards bookmarks. • 7: discards all comments, forms and multimedia. • 8: discards external cross references. • 9: discards document information and metadata. • 10: discards file attachments. • 11: discards private data of other applications. **Note** The <intlist> should be entered in the format "1,3,11" or "2-11" without any spaces.
-cl	*[-cl <string>]* *-cl <intlist>* *e.g.* *-cl "1"* *-cl "1,3,4"* *-cl "1-4"*	Clears up the streams, bookmarks or links. • 1: Use Flate to encode streams that are not encoded. • 2: In streams that use LZW encoding, use Flate instead. • 3: Remove invalid bookmarks. • 4: Remove invalid links. **Note** The <intlist> should be entered in the format "1,2,3,4" or "1-4" without any spaces.
-op	*[-op <string>]* *e.g.* *-op 123* *-op welcome*	Specifies the open password for the input file. Not required if the input file is not password protected. **Note** The output PDF file will retain the open password from the input file.

Option	Parameter	Description
-title	*<-title <string>>* *e.g.* *-title "Foxit PDF Toolkit User Manual"*	Sets title of PDF files.
-subject	*<-subject <string>>* *e.g.* *-subject "Foxit PDF Toolkit"*	Sets subject of PDF files.
-keywords	*[-keywords <string>]* *e.g.* *-keywords "Foxit"*	Sets keywords of PDF files.
-author	*[-author <string>]* *e.g.* *-author "Jessie"*	Sets author of PDF files.
-creator	*[-creator <string>]* *e.g.* *-creator "Foxit PhantomPDF"* *-creator "Foxit Reader"* *-creator "Microsoft® Word 2013"*	Sets creator of PDF files. **Note** It indicates the name of the application that created the source document from which a PDF is generated. For example, if the source document was created from Microsoft® *Word 2013, then you can set the creator to "*Microsoft® *Word 2013".*
-r	*[-r [integer]]* *e.g.* *-r* *-r 0* *-r 1* *-r 2* *…*	Specifies the number of layers to recurse when the input is a folder. • **-r 0 <-r>**: searches the full folders. • **-r 1**: searches only the current folder. • **-r 2**: searches the current folder and its sub-folders … **Note** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input PDF file or folder will be skipped if it is secured and the messages will be displayed.

Option	Parameter	Description
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2*	Specifies the number of CPU threads to use. The default value is 1.
-log	*[-log <string>]* *e.g.* *-log d:\a.log*	Writes log information into a logfile at the specified existing path.
-l	*[-l <integer>]* *e.g.* *-l 1* *-l 2* *-l 3* *-l 4*	Sets the log level. The default is 4. • **-l 1**: logs messages only concerning program crashes. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1. • **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <string> <string>]* *-register <code> <licensee>* *e.g.* *-register* 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foxit	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.
-help/-h	*[-help/-h]* *e.g.* *-help* *-h*	Prints the usage information.
-version/-v	*[-version/-v]* *e.g.* *-version* *-v*	Prints the version information.
-license	*[-license]* *e.g. -license*	Prints the license agreement.

3.5.3 Basic Usage

3.5.3.1 Input and Output

a) Input (-i)

- The input file should be a single PDF file or a folder. Users are not able to input multiple PDF files or folders, as well as a mixture composed of folders and PDF files. For example:

  -i c:\input\1.pdf (a single PDF file)
  -i c:\input (a single folder)

- It supports relative paths if the input file is in the current working folder. Users can input just the name of the PDF file or folder, instead of an absolute path. For example:

  -i test\2.pdf ("test\2.pdf" is in the current working folder)
  -i test ("test" folder is in the current working folder)

- It also supports wildcard characters, which are used to process multiple PDF files. For example:

  -i "c:\input*.pdf" (Only convert PDF files under "c:\input" folder)
  -i "test*.pdf" (Only convert PDF files under "test" folder)

  Note *When using wildcard characters in the input files, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

b) Output (-o)

- If the input is a single PDF file, you should specify the output path of a PDF file. If the input is a single folder, you should specify the output path of a folder. For example:

  -o d:\output\1_opt.pdf (a single PDF file)
  -o d:\output (a single folder)

  Note *The specified output path must already exist.*

- The output also supports relative paths if the specified output location is in the current working folder. Users can input just the name of the PDF file or output folder, instead of an absolute path. For example:

-o output\2_opt.pdf ("output" folder is in the current working folder)
-o output ("output" folder is in the current working folder)

Usage Examples

1) Optimize a single PDF file:

fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf

2) Optimize PDF files in a folder:

fpdfopt -i test -o output
fpdfopt -i c:\input -o d:\output
fpdfopt -i "c:\input*.pdf" -o d:\output

3.5.3.2 PDF/Page Settings

a) Color/grayscale images Downsampling (-dc)

- The optional argument (**-dc**) is used to downsample color/grayscale images in the input PDF file. By default, images will not be downsampled. For more details about this argument, please refer to section 3.5.2 "Command Line Summary".

Usage Example

1) Downsample all color/grayscale images with an original resolution higher than 300 dpi to a new resolution of 100 dpi using the Average Downsampling algorithm (-dc a 300 100)

fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -dc a 300 100
fpdfopt -i test -o output -dc a 300 100
fpdfopt -i c:\input -o d:\output -dc a 300 100

2) Downsample all color/grayscale images with an original resolution higher than 225 dpi to a new resolution of 150 dpi using the Bicubic Downsampling algorithm (-dc b, 225 and 150 are the default values)

fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -dc b
fpdfopt -i test -o output -dc b
fpdfopt -i c:\input -o d:\output -dc b

b) Color/grayscale images Compression (-cc)

- The optional argument (**-cc**) is used to compress color/grayscale images in the input PDF files with JPEG or JPEG2000 algorithm. For more details about this argument, please refer to section 3.5.2 "Command Line Summary".

Usage Example

1) Compress color/grayscale images with medium quality JEPG (-cc j medium)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -cc j medium
fpdfopt -i test -o output -cc j medium
fpdfopt -i c:\input -o d:\output -cc j medium
```

2) Compress color/grayscale images with maximum quality JEPG2000 (-cc j2 max)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -cc j2 max
fpdfopt -i test -o output -cc j2 max
fpdfopt -i c:\input -o d:\output -cc j2 max
```

c) Monochrome images Downsampling (-dm)

- The optional argument (**-dm**) is used to downsample monochrome images in the input PDF file. By default, images will not be downsampled. For more details about this argument, please refer to section 3.5.2 "Command Line Summary".

Usage Example

1) Downsample all monochrome images with an original resolution higher than 300 dpi to a new resolution of 100 dpi using the Average Downsampling algorithm (-dm a 300 100)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -dm a 300 100
fpdfopt -i test -o output -dm a 300 100
fpdfopt -i c:\input -o d:\output -dm a 300 100
```

2) Downsample all monochrome images with an original resolution higher than 450 dpi to a new resolution of 300 dpi using the Bicubic Downsampling algorithm (-dm b, 450 and 300 are the default values)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -dm b
```

fpdfopt -i test -o output -dm b
fpdfopt -i c:\input -o d:\output -dm b

d) Monochrome images Compression (-cm)

- The optional argument (**-cm**) is used to compress monochrome images in the input PDF file with JBIG2, CCITT Group 4, or Run Length algorithm. For more details about this argument, please refer to section 3.5.2 "Command Line Summary".

Usage Example

1) Compress monochrome images with lossless quality JBIG2 (-cm jbig2 lossless)

fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -cm jbig2 lossless
fpdfopt -i test -o output -cm jbig2 lossless
fpdfopt -i c:\input -o d:\output -cm jbig2 lossless

2) Compress monochrome images with Run Length (-cm runlength)

fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -cm runlength
fpdfopt -i test -o output -cm runlength
fpdfopt -i c:\input -o d:\output -cm runlength

e) Reduction (-rd)

- The optional argument (**-rd**) is used to optimize images only if there is a reduction in size.

Usage Example

1) Optimize images only if there is a reduction in size (-rd)

fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -rd
fpdfopt -i test -o output -rd
fpdfopt -i c:\input -o d:\output -rd

f) Unembedded fonts (-u)

- The optional argument (**-u**) is used to unembed all fonts in selected PDF documents.

Usage Example

1) Unembed all fonts in selected PDF documents (-u)

fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -u
fpdfopt -i test -o output -u
fpdfopt -i c:\input -o d:\output -u

g) Objects and user data Discard (-d)

- The optional argument (**-d**) is used to discard objects and user data in PDF files. For more details about this argument, please refer to section 3.5.2 "Command Line Summary".

Usage Example

1) Discard all form submission, import and reset actions (-d "1")

fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -d "1"
fpdfopt -i test -o output -d "1"
fpdfopt -i c:\input -o d:\output -d "1"

2) Flatten form fields, discard all JavaScript actions and discard bookmarks (-d "2,3,6")

fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -d "2,3,6"
fpdfopt -i test -o output -d "2,3,6"
fpdfopt -i c:\input -o d:\output -d "2,3,6"

h) Clear up (-cl)

- The optional argument (**-cl**) is used to clear up stream, bookmarks, or links. For more details about this argument, please refer to section 3.5.2 "Command Line Summary".

Usage Example

1) Use Flate to encode streams that are not encoded, and remove invalid bookmarks (-cl "1,3")

fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -cl "1,3"
fpdfopt -i test -o output -cl "1,3"
fpdfopt -i c:\input -o d:\output -cl "1,3"

i) Open Password (-op)

- The optional argument (**-op**) indicates the open password for a password-protected input PDF file. It is not required if the input file is not password protected.

  Note *The output PDF file will retain the open password from the input file.*

Usage Example

1) Specify the open password for a password-protected input PDF file (-op 123)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -op 123
fpdfopt -i test\2.pdf -o output\2_opt.pdf -op 123
```

2) Specify the open password for all input PDF files that have been protected with the same password (-op 123)

```
fpdfopt -i c:\input -o d:\output -op 123
fpdfopt -i test -o output -op 123
```

Note *It only supports typing one value for the argument (**-op**). Only files with the same open password can be processed together and files with different open password need to be processed separately.*

3.5.3.3 Document Metadata Settings

a) Title (-title)

- The optional argument (**-title**) is used to set title of PDF files.

Usage Example

1) Set document title to “Foxit PDF Toolkit User Manual” (-title “Foxit PDF Toolkit User Manual”)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -title “Foxit PDF Toolkit User Manual”
fpdfopt -i test -o output -title “Foxit PDF Toolkit User Manual”
fpdfopt -i “c:\input\*.pdf” -o d:\output -title “Foxit PDF Toolkit User Manual”
```

b) Subject (-subject)

- The optional argument (**-subject**) is used to set subject of PDF files.

Usage Example

1) Set document subject to “Foxit PDF Toolkit” (-subject “Foxit PDF Toolkit”)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -subject “Foxit PDF Toolkit”
fpdfopt -i test -o output -subject “Foxit PDF Toolkit”
fpdfopt -i “c:\input\*.pdf” -o d:\output -subject “Foxit PDF Toolkit”
```

c) Keywords (-keywords)

- The optional argument (**-keywords**) is used to set keywords of PDF files.

Usage Example

1) Set document keywords to “toolkit” (-keywords “toolkit”)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -keywords “toolkit”
fpdfopt -i test -o output -keywords “toolkit”
fpdfopt -i “c:\input\*.pdf” -o d:\output -keywords “toolkit”
```

d) Author (-author)

- The optional argument (**-author**) is used to set author of PDF files.

Usage Example

1) Set document author to “Jessie” (-author “Jessie”)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -author “Jessie”
fpdfopt -i test -o output -author “Jessie”
fpdfopt -i “c:\input\*.pdf” -o d:\output -author “Jessie”
```

e) Creator (-creator)

- The optional argument (**-creator**) is used to set file creation application information of PDF files.

Usage Example

1) Set document creator to “Foxit Reader” (-creator “Foxit Reader”)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -creator “Foxit Reader”
```

fpdfopt -i test -o output -creator “Foxit Reader”
fpdfopt -i “c:\input*.pdf” -o d:\output -creator “Foxit Reader”

3.5.3.4 Recursion Depth of Sub-folders

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard character like “c:\input*.pdf”. By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.5.2 “Command Line Summary”.

Usage Examples

1) Search the full folders (-r or -r 0)

fpdfopt -i test -o output -r
fpdfopt -i c:\input -o d:\output -r
fpdfopt -i test -o output -r 0
fpdfopt -i c:\input -o d:\output -r 0

2) Search only the current folder (-r 1)

fpdfopt -i test -o output -r 1
fpdfopt -i c:\input -o d:\output -r 1

Note *If you don’t use this argument, the current folder will be searched by default. For example:*

fpdfopt -i test -o output
fpdfopt -i c:\input -o d:\output

3) Search the current folder and its sub-folders (-r 2)

fpdfopt -i test -o output -r 2
fpdfopt -i c:\input -o d:\output -r 2

3.5.3.5 Multi-thread Support

- The optional argument (**-t**) indicates the number of threads that are used to speed up batch optimization by making full use of the CPU. By default, the number of the threads is 1.

Note *It is recommended that you set the value of the number according to your computer's CPU configuration.*

Usage Example

1) Set the number of threads to 3 (-t 3)

```
fpdfopt -i test -o output -t 3
fpdfopt -i c:\input -o d:\output -t 3
```

3.5.3.6 Other Optional Arguments

a) Log file (-log<logfile> -l<log level>)

- The optional argument (**-log**) indicates the path of logfile, where the log message is placed. The argument (**-l**) indicates the log level. It is valid only when (**-log**) is used. By default, the log level is 4. For more details about this argument, please refer to section 3.5.2 "Command Line Summary".

Usage Example

1) Save the log file to "d:\output\pdfoptimizer.log" and set the log level to 3 (-log d:\output\pdfoptimizer.log -l 3)

```
fpdfopt -i c:\input -o d:\output -log d:\output\pdfoptimizer.log -l 3
```

b) Register information (-register <code> <licensee>)

- The optional argument (**-register**) is used to register the command line tool. The **<code>** is the activation code from Foxit and **<licensee>** is the licensee name designated by the users.

Usage Example

5) Register the pdfoptimizer tool with the code "77505-010G0-G1000-XMQ8D-2CR7R-TPBEI" and the licensee "Foxit" (-register 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foixt)

```
fpdfopt -register 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foixt
```

c) License agreement (-license)

- The optional argument (**-license**) is used to print the license agreement.

Usage Example

1) Print the license information (-license)

```
fpdfopt -license
```

d) Version information (-version/-v)

- The optional argument (-version/-v) is used to print the version information.

Usage Example

1) Print the version information (-version/-v)

```
fpdfopt -version
fpdfopt -v
```

e) Help information (-help/-h)

- The optional argument (**-help/-h**) is used to print the usage information.

Usage Example

1) Print the usage information (-help/-h)

```
fpdfopt -help
fpdfopt -h
```

3.6 PDFRedactor

3.6.1 Basic Syntax

fpdfredact *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> <-mode <operation mode>>*
<-keyword <searchword>/<<pattern> <country>>>
[-character <singlepos>/<<startpos> <endpos>>] [-range <page range>] [-markcolor <R> <G> <B>]
[-tx <text>] [-ta <alignment>] [-font <fontname>] [-fs <fontsize>] [-fontcolor <R> <G> <B>] [-form]
[-op <password>] [-title <title>] [-subject <subject>] [-keywords <keywords>] [-author <author>]
[-creator <creator>] [-r [recursion]] [-t <thread>] [-log <logfile>] [-l <level>]

fpdfredact *-register <code> <licensee>*

fpdfredact *-license*

fpdfredact *-version/-v*

fpdfredact *-help/-h*

Note:

- <> required
- [ ] optional
- / mutually exclusive
- A space is needed between the command line argument and the value

Only the *<-i <srcfile/srcfolder>>*, *<-o <destfile/destfolder>>*, *<-mode <operation mode>>* and *<-keyword <searchword>/<<pattern> <country>>>* arguments are actually required. All others are optional, which are available for controlling the process as desired. The arguments could be given in any order. Full details on each are explained in the following section.

3.6.2 Command Line Summary

Note *For some arguments whose values are strings, users can choose whether to add quotation marks (" "). In the manual we have added notes where adding quotation marks (" ") is required.*

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i c:\input\1.pdf* *-i c:\input* *-i "c:\input*.pdf"*	Specifies the input file to be redacted. ▪ The input string can be the name of a single PDF file or a folder. ▪ The file name can contain the wildcard character (*). For example, use *.pdf to

Option	Parameter	Description
		include all PDF files in a given folder. **Note** Wildcard character (*.*) is currently not supported.
-o	*<-o <string>>* *e.g.* *-o d:\output\1_reda.pdf* *-o d:\output*	Specifies the path of the output PDF file or folder. ▪ If the input is a PDF file, the output should be a single PDF file, (*e.g. -o d:\output\1_reda.pdf*). ▪ If the input is a folder, the output should be a folder, (*e.g. -o d:\output*). **Note** The specified output path must already exist.
-mode	*<-mode <integer>>* *e.g.* *-mode 1* *-mode 2*	Specifies the mode of the keyword to be searched. • **-mode 1**: sets the mode of the keyword to a word or phrase. • **-mode 2**: sets the mode of the keyword to a pattern.
-keyword	*<-keyword <string>/<<integer> <integer>>>* *<-keyword <word/phrase> / <<pattern> <country>>>* *e.g.* *-keyword "Foxit"* *-keyword "Foxit PDF Toolkit"* *-keyword 1 1* *-keyword 1 2* *-keyword 2 3* *-keyword 3 4* *-keyword 4 5* *-keyword 5 8* *-keyword 5 10* *…*	Specifies the search keyword. There are two types of keyword and the "**-mode**" argument determines which one to use. ▪ **-keyword <word/phrase>** is used when **-mode 1** is set. And **<word/phrase>** is a string with quotation marks. ▪ **-keyword <pattern> <country>** is used when **-mode 2** is set. **<pattern>** value could be: **1:** Phone Number **2:** Date **3:** Social Security Number **4:** Email Address **5:** Credit Card Number **<country>** value could be: **1:** EN_US **2:** China **3:** China_TW&HK **4:** French **5:** German

Option	Parameter	Description
		6: Japan **7:** Korean **8:** Brazil **9:** Spain **10:** Turkey **Note** If the keyword contains special characters like quotation marks and slashes, you should add the escape character (\) before them.
-character	*[-character <integer>/<<integer> <integer>>]* *-character <singlepos>/ <<startpos> <endpos>>* *e.g.* *-character 3* *-character 1 4*	Specifies the characters of the keyword you want to redact. By default, all the characters of the keyword will be redacted. ▪ **-character <singlepos>**: redacts only the ***singlepos*** character of the keyword. For example, *-character 3*: redacts only the third character of the keyword. ▪ **-character <startpos> <endpos>**: redacts the characters of the keyword in the range from ***startpos*** to ***endpos***. For example, *-character 1 4*: redacts the first character to the fourth character of the keyword.
-range	*-rang <String>* *e.g.* *-range "1,5,9"* *-range "all"* *-range "even"* *-range "odd"* *-range "2-10,30"* *-range "10,50-"* *-range "odd,100-"*	Specifies a page range to apply redaction. By default, all of the pages will apply redaction. ▪ Applies redaction to pages 1,5, and 9: **-range "1,5,9"** ▪ Applies redaction to all of the pages: **-range "all"** ▪ Applies redaction to all even pages: **-range "even"** ▪ Applies redaction to all odd pages: **-range "odd"** ▪ Applies redaction to pages in the range from 2-10 and page 30:

Option	Parameter	Description
		-range "2-10,30" ▪ Applies redaction to page 10 and all pages in the range from 50 to the last page: **-range "10,50-"** ▪ Applies redaction to all odd pages and all pages in the range from 100 to the last page: **-range "odd,100-"**
-markcolor	*[-markcolor <integer> <integer> <integer>]* *-markcolor <R> <G> <B>* *e.g.* *-markcolor 0 0 0* *-markcolor 255 255 0* *-markcolor 255 255 255* *…*	Specifies the fill color used to mark for redaction with the RGB color model. By default, the fill color is black. Allowable range of the values for each RGB component is 0-255.
-tx	*[-tx <string>]* *-tx <text>* *e.g.* *-tx "secret"*	Overlays redaction marks with the specified text. **Note** If the keyword contains special characters like quotation marks and slashes, you should add the escape character (\) before them.
-ta	*[-ta <integer>]* *-ta <alignment>* *e.g.* *-ta 1* *-ta 2* *-ta 3*	Sets the alignment of the text designated by the **-tx** option. The default value is 1. • **-ta 1**: aligns the text left. • **-ta 2**: aligns the text center. • **-ta 3**: aligns the text right. **Note** The (**-ta**) is valid only when (**-tx**) is used.
-font	*[-font <string>]* *-font <fontname>* *e.g.* *-font "Calibri"* *-font "Helvetica"* *-font "Arial"* *…*	Sets the font style of the text designated by the **-tx** option. **Note** ▪ The font style should be installed in a local environment, otherwise the default font style will be used. ▪ The (**-font**) is valid only when (**-tx**) is used.

Option	Parameter	Description
-fs	*[-fs <integer>]* *-fs <fontsize>* *e.g.* *-fs 8* *-fs 11* *-fs 72*	Sets the font size of the text designated by the **-tx** option. Default value: 9. Allowable range: 8-72. **Note** The (**-fs**) is valid only when (**-tx**) is used.
-fontcolor	*[-fontcolor <integer> <integer> <integer>]* *-fontcolor <R> <G> <B>* *e.g.* *-fontcolor 0 0 0* *-fontcolor 255 255 0* *-fontcolor 255 255 255* *…*	Specifies the font color of the text designated by the **-tx** option with the RGB color model. By default, the font color is green. Allowable range of the values for each RGB component is 0-255. **Note** The (**-fontcolor**) is valid only when (**-tx**) is used.
-form	*[-form]*	If this argument is set, both PDF forms and main body of text will be searched for the specified keyword, otherwise only main body of text will be searched. **Note** If the keyword in the PDF form is generated by JavaScript, it will not be searched.
-op	*[-op<string>]*	Specifies the open password for the input file. Not required if the input file is not password protected. **Note** The output PDF file will retain the open password from the input file.
-title	*<-title <string>>* *e.g.* *-title "Foxit PDF Toolkit User Manual"*	Sets title of PDF files.
-subject	*<-subject <string>>* *e.g.* *-subject "Foxit PDF Toolkit"*	Sets subject of PDF files.
-keywords	*[-keywords <string>]* *e.g.* *-keywords "Foxit"*	Sets keywords of PDF files.

Option	Parameter	Description
-author	*[-author <string>]* *e.g.* *-author "Jessie"*	Sets author of PDF files.
-creator	*[-creator <string>]* *e.g.* *-creator "Foxit PhantomPDF"* *-creator "Foxit Reader"* *-creator "Microsoft® Word 2013"*	Sets creator of PDF files. **Note** It indicates the name of the application that created the source document from which a PDF is generated. For example, if the source document was created from Microsoft® *Word 2013, then you can set the creator to "*Microsoft® *Word 2013".*
-r	*[-r [integer]]* *e.g.* *-r* *-r 0* *-r 1* *-r 2* …	Specifies the number of layers to recurse when the input is a folder. • **-r 0 <-r>**: searches the full folders. • **-r 1**: searches only the current folder. • **-r 2**: searches the current folder and its sub-folders. … **Note** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input PDF file or folder will be skipped if it is secured and the messages will be displayed.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2*	Specifies the number of CPU threads to use. The default value is 1.
-log	*[-log <string>]* *e.g.* *-log d:\a.log*	Writes log information into a logfile at the specified existing path.
-l	*[-l <integer>]* *e.g.*	Sets the log level. The default is 4. • **-l 1**: logs messages only concerning program

Option	Parameter	Description
	-l 1 *-l 2* *-l 3* *-l 4*	crashes. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1. • **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>* *e.g.* *-register* 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foxit	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.
-help/-h	*[-help]/[-h]* *e.g.* *-help* *-h*	Prints the usage information.
-version/-v	*[-version]/[-v]* *e.g.* *-version* *-v*	Prints the version information.
-license	*[-license]* *e.g.* *-license*	Prints the license agreement.

3.6.3 Basic Usage

3.6.3.1 Required Arguments

a) Input (-i)

➢ The input file should be a single PDF file or a folder. Users are not able to input multiple PDF files or folders, as well as a mixture composed of folders and PDF files. For example:

-i c:\input\1.pdf (a single PDF file)

-i c:\input (a single folder)

- It supports relative paths if the input file is in the current working folder. Users can input just the name of the PDF file or folder, instead of an absolute path. For example:

-i test\2.pdf ("test\2.pdf" is in the current working folder)
-i test ("test" folder is in the current working folder)

- It also supports wildcard characters, which are used to process multiple PDF files. For example:

-i "c:\input*.pdf" (Only convert PDF files under "c:\input" folder)
-i "test*.pdf" (Only convert PDF files under "test" folder)

Note *When using wildcard characters in the input files, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

b) Output (-o)

- If the input is a single PDF file, you should specify the output path of a PDF file. If the input is a single folder, you should specify the output path of a folder. For example:

-o d:\output\1_reda.pdf (a single PDF file)
-o d:\output (a single folder)

Note *The specified output path must already exist.*

- The output also supports relative paths if the specified output location is in the current working folder. Users can input just the name of the output PDF file or folder, instead of an absolute path. For example:

-o output\2_reda.pdf ("output" folder is in the current working folder)
-o output ("output" folder is in the current working folder)

c) Operation Mode (-mode)

- The operation mode argument (-mode) is required in the command line for specifying the mode of the keyword to be searched. There are two types of keywords: one is a word or phrase, and the other is a pattern. For example,

-mode 1 (keyword is a word or phrase)
-mode 2 (keyword is a pattern)

d) Search Keyword (-keyword)

- The search keyword argument (**-keyword**) is required in the command line and its value is dependent on the setting of the argument (**-mode**). For more details about this argument, please refer to 3.6.2 "Command Line Summary". The **-mode** and **-keyword** arguments are grouped together. For example,

  -mode 1 -keyword "Foxit"
  -mode 1 -keyword "Foxit Software"
  -mode 2 -keyword 1 2
  -mode 2 -keyword 2 8

Usage Examples

1) Redact(Remove) the sensitive content "Foxit" from PDF file(s):

   fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit"
   fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 1 -keyword "Foxit"
   fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit"
   fpdfredact -i test -o output -mode 1 -keyword "Foxit"
   fpdfredact -i "c:\input*.pdf" -o d:\output -mode 1 -keyword "Foxit"

2) Redact(Remove) the sensitive content "Phone Number" pattern in "En_US" country from PDF file(s):

   fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 2 -keyword 1 1
   fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1
   fpdfredact -i c:\input -o d:\output -mode 2 -keyword 1 1
   fpdfredact -i test -o output -mode 2 -keyword 1 1
   fpdfredact -i "c:\input*.pdf" -o d:\output -mode 2 -keyword 1 1

3.6.3.2 Redaction Settings

a) Redaction range of the Keyword (-character)

- The optional argument (**-character**) is used to specify the characters of the keyword you want to redact. If this argument is not set, all the characters of the keyword will be redacted by default. For more details about this argument, please refer to section 3.6.2 "Command Line Summary".

Usage Example

1) Redact only the third character of the keyword (-character 3)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -character 3
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -character 3
fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit Software” -character 3
fpdfredact -i test -o output -mode 2 -keyword 1 1 -character 3
fpdfredact -i “c:\input\*.pdf” -o d:\output -mode 2 -keyword 2 6 -character 3
```

2) Redacts the first character to the fourth character of the keyword (-character 1 4)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -character 1 4
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -character 1 4
fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit Software” -character 1 4
fpdfredact -i test -o output -mode 2 -keyword 1 1 -character 1 4
fpdfredact -i “c:\input\*.pdf” -o d:\output -mode 2 -keyword 2 6 -character 1 4
```

b) Page range to apply redaction (-range)

- The optional argument (**-range**) is used to specify a page range to apply redaction. If this argument is not set, all of the pages will apply redaction. For more details about this argument, please refer to section 3.6.2 “Command Line Summary”.

Usage Example

1) Apply redaction to pages 2,3 and 8 (-range “2,3,8”)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -range “2,3,8”
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -range “2,3,8”
fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit Software” -range “2,3,8”
fpdfredact -i test -o output -mode 2 -keyword 1 1 -range “2,3,8”
fpdfredact -i “c:\input\*.pdf” -o d:\output -mode 2 -keyword 2 6 -range “2,3,8”
```

2) Apply redaction to all even pages (-range “even”)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -range “even”
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -range “even”
fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit Software” -range “even”
fpdfredact -i test -o output -mode 2 -keyword 1 1 -range “even”
```

fpdfredact -i “c:\input*.pdf” -o d:\output -mode 2 -keyword 2 6 -range “even”

3) Apply redaction to pages in the range from 2-10 and page 30 (-range “2-10,30”)

fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -range “2-10,30”
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -range “2-10,30”
fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit Software” -range “2-10,30”
fpdfredact -i test -o output -mode 2 -keyword 1 1 -range “2-10,30”
fpdfredact -i “c:\input*.pdf” -o d:\output -mode 2 -keyword 2 6 -range “2-10,30”

4) Apply redaction to all odd pages and all pages in the range from 100 to the last page (-range “odd,100-”)

fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -range “odd,100-”
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -range “odd,100-”
fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit Software” -range “odd,100-”
fpdfredact -i test -o output -mode 2 -keyword 1 1 -range “odd,100-”
fpdfredact -i “c:\input*.pdf” -o d:\output -mode 2 -keyword 2 6 -range “odd,100-”

c) Fill color for redaction marks (-markcolor)

- The optional argument (**-markcolor**) is used to specify the fill color used to mark for redaction with the RGB color model. If this argument is not set, the fill color is black by default. The allowable range of the values for each RGB component is from 0 to 255.

Usage Example

1) Set the fill color to blue for the redaction marks (-markcolor 0 0 255)

fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -markcolor 0 0 255
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -markcolor 0 0 255
fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit Software” -markcolor 0 0 255
fpdfredact -i test -o output -mode 2 -keyword 1 1 -markcolor 0 0 255
fpdfredact -i “c:\input*.pdf” -o d:\output -mode 2 -keyword 2 6 -markcolor 0 0 255

d) Overlay text (-tx)

- The optional argument (**-tx**) is used to add overlay text to redaction marks. Overlay text appears on top of redaction marks, which is useful for sensitive content that needs to be removed after redaction. Users can specify custom text to appear over the redaction marks.

Usage Example

1) Use overlay text “secret” to appear on sensitive text “Foxit” selected for redaction (-tx “secret”)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -tx “secret”
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -tx “secret”
fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit Software” -tx “secret”
fpdfredact -i test -o output -mode 2 -keyword 1 1  -tx “secret”
fpdfredact -i “c:\input\*.pdf” -o d:\output -mode 2 -keyword 2 6 -tx “secret”
```

e) Text alignment (-ta)

- The optional argument (**-ta**) is used to set alignment of the overlay text designated by the **-tx** option. It is valid only when (**-tx**) is used. The default value is 1, which will be left aligned.

Usage Example

1) Align the text to the left (-ta 1)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -tx “secret” -ta 1
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -tx “secret” -ta 1
fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit Software” -tx “secret” -ta 1
fpdfredact -i test -o output -mode 2 -keyword 1 1  -tx “secret” -ta 1
fpdfredact -i “c:\input\*.pdf” -o d:\output -mode 2 -keyword 2 6 -tx “secret” -ta 1
```

2) Align the text to the center (-ta 2)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -tx “secret” -ta 2
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -tx “secret” -ta 2
fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit Software” -tx “secret” -ta 2
fpdfredact -i test -o output -mode 2 -keyword 1 1  -tx “secret” -ta 2
fpdfredact -i “c:\input\*.pdf” -o d:\output -mode 2 -keyword 2 6 -tx “secret” -ta 2
```

3) Align the text to the right (-ta 3)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -tx “secret” -ta 3
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -tx “secret” -ta 3
fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit Software” -tx “secret” -ta 3
fpdfredact -i test -o output -mode 2 -keyword 1 1  -tx “secret” -ta 3
fpdfredact -i “c:\input\*.pdf” -o d:\output -mode 2 -keyword 2 6 -tx “secret” -ta 3
```

f) Text font (-font)

- The optional argument (**-font**) is used to set the font style of the overlay text designated by the **-tx** option. It is valid only when (**-tx**) is used.

  Note *The font style should be installed in a local environment, otherwise the default font style will be used.*

Usage Example

1) Set the font of the overlay text to "Calibri" (-font "Calibri")

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -tx "secret" -font "Calibri"
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -tx "secret" -font "Calibri"
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -tx "secret" -font "Calibri"
fpdfredact -i test -o output -mode 2 -keyword 1 1  -tx "secret" -font "Calibri"
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -tx "secret" -font "Calibri"
```

g) Text font size (-fs)

- The optional argument (**-fs**) is used to set the font size of the overlay text designated by the **-tx** option. It is valid only when (**-tx**) is used. The default value is set to 9, and the allowable range is from 8 to 72.

Usage Example

1) Set the font size of the overlay text to 12 (-fs 12)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -tx "secret" -fs 12
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -tx "secret" -fs 12
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -tx "secret" -fs 12
fpdfredact -i test -o output -mode 2 -keyword 1 1  -tx "secret" -fs 12
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -tx "secret" -fs 12
```

h) Text font color (-fontcolor)

- The optional argument (**-fontcolor**) is used to set the font color of the overlay text designated by the **-tx** option with the RGB color model. It is valid only when (**-tx**) is used. By default, the font color is green. The allowable range of the values for each RGB component is from 0 to 255.

Usage Example

1) Set the font color of the overlay text to yellow (-fontcolor 255 255 0)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -tx “xx” -fontcolor 255 255 0
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -tx “xx” -fontcolor 255 255 0
fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit Software” -tx “xx” -fontcolor 255 255 0
fpdfredact -i test -o output -mode 2 -keyword 1 1  -tx “xx” -fontcolor 255 255 0
fpdfredact -i “c:\input\*.pdf” -o d:\output -mode 2 -keyword 2 6 -tx “xx” -fontcolor 255 255 0
```

i) Form (-form)

- The optional argument (**-form**) is used to search the specified keyword in both PDF forms and the main body of text. If this argument is not set, only main body of text will be searched.

  Note *If the keyword in the PDF form is generated by JavaScript, it will not be searched.*

Usage Example

1) Search the specified keyword in both PDF forms and the main body of text (-form)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -form
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -form
fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit Software” -form
fpdfredact -i test -o output -mode 2 -keyword 1 1 -form
fpdfredact -i “c:\input\*.pdf” -o d:\output -mode 2 -keyword 2 6 -form
```

3.6.3.3 Open Password

- The optional argument (**-op**) indicates the open password for a password-protected input PDF file. It is not required if the input file is not password protected.

  Note *The output PDF file will retain the open password from the input file.*

Usage Example

1) Specify the open password for a password-protected input PDF file (-op 123)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -op 123
```

fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -op 123

2) Specify the open password for all input PDF files that have been protected with the same password (-op 123)

fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit” -op 123
fpdfredact -i test -o output -mode 2 -keyword 1 1 -op 123

Note *It only supports typing one value for the argument (**-op**). Only files with the same open password can be processed together and files with different open password need to be processed separately.*

3.6.3.4 Document Metadata Settings

a) Title (-title)

- The optional argument (**-title**) is used to set title of PDF files.

Usage Example

1) Set document title to “Foxit PDF Toolkit User Manual” (-title “Foxit PDF Toolkit User Manual”)

fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -title “Foxit PDF Toolkit User Manual”
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -title “Foxit PDF Toolkit User Manual”
fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit Software” -title “Foxit PDF Toolkit User Manual”
fpdfredact -i test -o output -mode 2 -keyword 1 1 -title “Foxit PDF Toolkit User Manual”
fpdfredact -i “c:\input*.pdf” -o d:\output -mode 2 -keyword 2 6 -title “Foxit PDF Toolkit User Manual”

b) Subject (-subject)

- The optional argument (**-subject**) is used to set subject of PDF files.

Usage Example

1) Set document subject to “Foxit PDF Toolkit” (-subject “Foxit PDF Toolkit”)

fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -subject “Foxit PDF Toolkit”
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -subject “Foxit PDF Toolkit”
fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit Software” -subject “Foxit PDF Toolkit”
fpdfredact -i test -o output -mode 2 -keyword 1 1 -subject “Foxit PDF Toolkit”
fpdfredact -i “c:\input*.pdf” -o d:\output -mode 2 -keyword 2 6 -subject “Foxit PDF Toolkit”

c) Keywords (-keywords)

- The optional argument (**-keywords**) is used to set keywords of PDF files.

Usage Example

1) Set document keywords to “toolkit” (-keywords “toolkit”)

fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -keywords “toolkit”
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -keywords “toolkit”
fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit Software” -keywords “toolkit”
fpdfredact -i test -o output -mode 2 -keyword 1 1 -keywords “toolkit”
fpdfredact -i “c:\input*.pdf” -o d:\output -mode 2 -keyword 2 6 -keywords “toolkit”

d) Author (-author)

- The optional argument (**-author**) is used to set author of PDF files.

Usage Example

1) Set document author to “Jessie” (-author “Jessie”)

fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -author “Jessie”
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -author “Jessie”
fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit Software” -author “Jessie”
fpdfredact -i test -o output -mode 2 -keyword 1 1 -author “Jessie”
fpdfredact -i “c:\input*.pdf” -o d:\output -mode 2 -keyword 2 6 -author “Jessie”

e) Creator (-creator)

- The optional argument (**-creator**) is used to set file creation application information of PDF files.

Usage Example

1) Set document creator to “Foxit Reader” (-creator “Foxit Reader”)

fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword “Foxit” -creator “Foxit Reader”
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -creator “Foxit Reader”
fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit Software” -creator “Foxit Reader”
fpdfredact -i test -o output -mode 2 -keyword 1 1 -creator “Foxit Reader”
fpdfredact -i “c:\input*.pdf” -o d:\output -mode 2 -keyword 2 6 -creator “Foxit Reader”

3.6.3.5 Recursion Depth of Sub-folders

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard character like “c:\input*.pdf”. By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.6.2 “Command Line Summary”.

Usage Examples

1) Search the full folders (-r or -r 0)

fpdfredact -i test -o output -mode 1 -keyword “Foxit” -r
fpdfredact -i c:\input -o d:\output -mode 2 -keyword 2 5 -r
fpdfredact -i c:\input*.pdf -o d:\output -mode 2 -keyword 3 6 -r
fpdfredact -i test -o output -mode 1 -keyword “Foxit” -r 0
fpdfredact -i c:\input -o d:\output -mode 2 -keyword 2 5 -r 0
fpdfredact -i “c:\input*.pdf” -o d:\output -mode 2 -keyword 3 6 -r 0

2) Search only the current folder (-r 1)

fpdfredact -i test -o output -mode 1 -keyword “Foxit” -r 1
fpdfredact -i c:\input -o d:\output -mode 2 -keyword 2 5 -r 1
fpdfredact -i “c:\input*.pdf” -o d:\output -mode 2 -keyword 3 6 -r 1

Note *If you don’t use this argument, the current folder will be searched by default. For example:*

fpdfredact -i test -o output -mode 1 -keyword “Foxit”
fpdfredact -i c:\input -o d:\output -mode 2 -keyword 2 5
fpdfredact -i “c:\input*.pdf” -o d:\output -mode 2 -keyword 3 6

3) Search the current folder and its sub-folders (-r 2)

fpdfredact -i test -o output -mode 1 -keyword “Foxit” -r 2
fpdfredact -i c:\input -o d:\output -mode 2 -keyword 2 5 -r 2
fpdfredact -i “c:\input*.pdf” -o d:\output -mode 2 -keyword 3 6 -r 2

3.6.3.6 Multi-thread Support

- The optional argument (**-t**) indicates the number of threads that are used to speed up batch programming by making full use of the CPU. By default, the number of threads is 1.

  Note *It is recommended that you set the value of the number according to your computer’s CPU configuration.*

Usage Example

1) Set the number of threads to 3 (-t 3)

fpdfredact -i test -o output -mode 1 -keyword “Foxit” -t 3
fpdfredact -i c:\input -o d:\output -mode 2 -keyword 2 5 -t 3
fpdfredact -i “c:\input*.pdf” -o d:\output -mode 2 -keyword 3 6 -t 3

3.6.3.7 Other Optional Arguments

a) Log file (-log<logfile> -l<log level>)

- The optional argument (**-log**) indicates the path of logfile, where the log message is placed. The argument (**-l**) indicates the log level. It is valid only when (**-log**) is used. By default, the log level is 4. For more details about this argument, please refer to section 3.6.2 “Command Line Summary”.

Usage Example

1) Save the log file to “d:\output\pdfredactor.log” and set the log level to 3 (-log d:\output\pdfredactor.log -l 3)

fpdfredact -i c:\input -o d:\output -mode 1 -keyword “Foxit” -log d:\output\pdfredactor.log -l 3

b) Register information (-register <code> <licensee>)

- The optional argument (**-register**) is used to register the command line tool. The **<code>** is the activation code from Foxit and the **<licensee>** is the licensee name designated by the users.

Usage Example

6) Register the pdfredactor tool with the code "77505-010G0-G1000-XMQ8D-2CR7R-TPBEI" and the licensee "Foxit" (-register 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foixt)

```
fpdfredact -register 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foixt
```

c) License agreement (-license)

- The optional argument (**-license**) is used to print the license agreement.

Usage Example

1) Print the license agreement (-license)

```
fpdfredact -license
```

d) Version information (-version/-v)

- The optional argument (-version/-v) is used to print the version information.

Usage Example

1) Print the version information (-version/-v)

```
fpdfredact -version
fpdfredact -v
```

e) Help information (-help/-h)

- The optional argument (**-help/-h**) is used to print the usage information.

Usage Example

1) Print the usage information (-help/-h)

```
fpdfredact -help
fpdfredact -h
```

3.7 PDFMetadata

3.7.1 Basic Syntax

fpdfmeta *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> [-title <title>] [-subject <subject>]*
[-keywords <keywords>] [-author <author>] [-creator <creator>] [-op <password>]
[-r [recursion]] [-t <thread>] [-log <logfile>] [-l <level>]
fpdfmeta *<-i <srcfile>> <-properties> [-t 1] [-op <password>] [-log <logfile>] [-l <level>]*
fpdfmeta *-register <code> <licensee>*
fpdfmeta *-license*
fpdfmeta *-version/-v*
fpdfmeta *-help/-h*

Note:

- <> required
- [ ] optional
- / mutually exclusive
- A space is needed between the command line argument and the value

If you want to set document metadata for PDF files, you should set not only *<-i <srcfile/srcfolder>>* and *<-o <destfile/destfolder>>* arguments, but also at least one of the following arguments (-title, -subject, -keywords,-author and -creator). If you just want to view PDF metadata information, the *<-i <srcfile>>* and *<-properties>* arguments are required.
All others are optional, which are available for controlling the process as desired. The arguments could be given in any order. Full details on each are explained in the following section.

3.7.2 Command Line Summary

Note *For some arguments whose values are strings, users can choose whether to add quotation marks (" "). In the manual we have added notes where adding quotation marks (" ") is required.*

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i c:\input\1.pdf* *-i c:\input* *-i "c:\input*.pdf"*	Specifies the input file to be processed. ▪ The input string can be the name of a single PDF file or a folder. ▪ The file name can contain the wildcard character (*). For example, use *.pdf to

Option	Parameter	Description
		include all PDF files in a given folder. **Note** Wildcard character (*.*) is currently not supported.
-o	*<-o <string>>* *e.g.* *-o d:\output\1_meta.pdf* *-o d:\output*	Specifies the path of the output PDF file or folder. ▪ If the input is a PDF file, the output should be a single PDF file, (*e.g. -o d:\output\1_meta.pdf*). ▪ If the input is a folder, the output should be a folder, (*e.g. -o d:\output*). **Note** The specified output path must already exist.
-title	*<-title <string>>* *e.g.* *-title "Foxit PDF Toolkit User Manual"*	Sets title of PDF files.
-subject	*<-subject <string>>* *e.g.* *-subject "Foxit PDF Toolkit"*	Sets subject of PDF files.
-keywords	*[-keywords <string>]* *e.g.* *-keywords "Foxit"* *-keywords "img2pdf office2pdf pdfwatermark pdfmetadata"*	Sets keywords of PDF files.
-author	*[-author <string>]* *e.g.* *-author "Jessie"*	Sets author of PDF files.
-creator	*[-creator <string>]* *e.g.* *-creator "Foxit PhantomPDF"* *-creator "Foxit Reader"* *-creator "Microsoft® Word 2013"*	Sets creator of PDF files. **Note** It indicates the name of the application that created the source document from which a PDF is generated. For example, if the source document was created from Microsoft® *Word 2013, then you can set the creator to "*Microsoft® *Word 2013".*

Option	Parameter	Description
-properties	*<-properties>* *e.g.* *-properties*	Views metadata information of the PDF file designated by the **-i** option, such as Title, Subject, Author, Keywords and Creator. **Note** The argument (**-properties**) can be only used with **-i**, **-t**, **-op**, **-log** and **-l** options. The value of **-i** must be a PDF file, and the value of **-t** must be 1 if it is set.
-op	*[-op<string>]* *e.g.* *-op 123* *-op welcome*	Specifies the open password for the input file. Not required if the input file is not password protected. **Note** The output PDF file will retain the open password from the input file.
-r	*[-r [integer]]* *e.g.* *-r* *-r 0* *-r 1* *-r 2* …	Specifies the number of layers to recurse when the input is a folder. • **-r 0 <-r>**: searches the full folders. • **-r 1**: searches only the current folder. • **-r 2**: searches the current folder and its sub-folders … **Note** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input PDF file or folder will be skipped if it is secured and the messages will be displayed.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2*	Specifies the number of CPU threads to use. The default value is 1.
-log	*[-log <string>]* *e.g.* *-log d:\a.log*	Writes log information into a logfile at the specified existing path.

Option	Parameter	Description
-l	*[-l <integer>]* *e.g.* *-l 1* *-l 2* *-l 3* *-l 4*	Sets the log level. The default is 4. • **-l 1**: logs messages only concerning program crashes. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1. • **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>* *e.g.* *-register* 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foxit	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.
-help/-h	*[-help]/[-h]* *e.g.* *-help* *-h*	Prints the usage information.
-version/-v	*[-version]/[-v]* *e.g.* *-version* *-v*	Prints the version information.
-license	*[-license]* *e.g.* *-license*	Prints the license agreement.

3.7.3 Basic Usage

3.7.3.1 Input and Output

a) Input (-i)

- The input file should be a single PDF file or a folder. Users are not able to input multiple PDF files or folders, as well as a mixture composed of folders and PDF files. For example:

  -i c:\input\1.pdf (a single PDF file)
  -i c:\input (a single folder)

- It supports relative paths if the input file is in the current working folder. Users can input just the name of the PDF file or folder, instead of an absolute path. For example:

  -i test\2.pdf ("test\2.pdf" is in the current working folder)
  -i test ("test" folder is in the current working folder)

- It also supports wildcard characters, which are used to process multiple PDF files. For example:

  -i "c:\input*.pdf" (Only convert PDF files under "c:\input" folder)
  -i "test*.pdf" (Only convert PDF files under "test" folder)

  Note *When using wildcard characters in the input files, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

b) Output (-o)

- If the input is a single PDF file, you should specify the output path of a PDF file. If the input is a single folder, you should specify the output path of a folder. For example:

  -o d:\output\1_meta.pdf (a single PDF file)
  -o d:\output (a single folder)

  Note *The specified output path must already exist.*

- The output also supports relative paths if the specified output location is in the current working folder. Users can input just the name of the output PDF file or folder, instead of an absolute path. For example:

  -o output\2_meta.pdf ("output" folder is in the current working folder)
  -o output ("output" folder is in the current working folder)

3.7.3.2 Document Metadata Settings

a) Title (-title)

- The optional argument (**-title**) is used to set title of PDF files.

Usage Example

1) Set document title to “Foxit PDF Toolkit User Manual” (-title “Foxit PDF Toolkit User Manual”)

```
fpdfmeta -i c:\input\1.pdf -o d:\output\1_meta.pdf -title “Foxit PDF Toolkit User Manual”
fpdfmeta -i test\2.pdf -o output\2_meta.pdf -title “Foxit PDF Toolkit User Manual”
fpdfmeta -i test -o output -title “Foxit PDF Toolkit User Manual”
fpdfmeta -i c:\input -o d:\output -title “Foxit PDF Toolkit User Manual”
fpdfmeta -i “c:\input\*.pdf” -o d:\output -title “Foxit PDF Toolkit User Manual”
```

b) Subject (-subject)

- The optional argument (**-subject**) is used to set subject of PDF files.

Usage Example

1) Set document subject to “Foxit PDF Toolkit” (-subject “Foxit PDF Toolkit”)

```
fpdfmeta -i c:\input\1.pdf -o d:\output\1_meta.pdf -subject “Foxit PDF Toolkit”
fpdfmeta -i test\2.pdf -o output\2_meta.pdf -subject “Foxit PDF Toolkit”
fpdfmeta -i test -o output -subject “Foxit PDF Toolkit”
fpdfmeta -i c:\input -o d:\output -subject “Foxit PDF Toolkit”
fpdfmeta -i “c:\input\*.pdf” -o d:\output -subject “Foxit PDF Toolkit”
```

c) Keywords (-keywords)

- The optional argument (**-keywords**) is used to set keywords of PDF files.

Usage Example

1) Set document keywords to “Foxit” (-keywords “Foxit”)

```
fpdfmeta -i c:\input\1.pdf -o d:\output\1_meta.pdf -keywords “Foxit”
fpdfmeta -i test\2.pdf -o output\2_meta.pdf -keywords “Foxit”
```

fpdfmeta -i test -o output -keywords “Foxit”
fpdfmeta -i c:\input -o d:\output -keywords “Foxit”
fpdfmeta -i “c:\input*.pdf” -o d:\output -keywords “Foxit”

2) Set document keywords to “image2pdf pdfmetadata pdfwatermark” (-keywords “image2pdf pdfmetadata pdfwatermark”)

fpdfmeta -i c:\input\1.pdf -o d:\output\1_meta.pdf -keywords “image2pdf pdfmetadata pdfwatermark”
fpdfmeta -i test\2.pdf -o output\2_meta.pdf -keywords “image2pdf pdfmetadata pdfwatermark”
fpdfmeta -i test -o output -keywords “image2pdf pdfmetadata pdfwatermark”
fpdfmeta -i c:\input -o d:\output -keywords “image2pdf pdfmetadata pdfwatermark”
fpdfmeta -i “c:\input*.pdf” -o d:\output -keywords “image2pdf pdfmetadata pdfwatermark”

d) Author (-author)

- The optional argument (**-author**) is used to set author of PDF files.

Usage Example

1) Set document author to “Jessie” (-author “Jessie”)

fpdfmeta -i c:\input\1.pdf -o d:\output\1_meta.pdf -author “Jessie”
fpdfmeta -i test\2.pdf -o output\2_meta.pdf -author “Jessie”
fpdfmeta -i test -o output -author “Jessie”
fpdfmeta -i c:\input -o d:\output -author “Jessie”
fpdfmeta -i “c:\input*.pdf” -o d:\output -author “Jessie”

e) Creator (-creator)

- The optional argument (**-creator**) is used to set file creation application information of PDF files.

Usage Example

1) Set document creator to “Foxit Reader” (-creator “Foxit Reader”)

fpdfmeta -i c:\input\1.pdf -o d:\output\1_meta.pdf -creator “Foxit Reader”
fpdfmeta -i test\2.pdf -o output\2_meta.pdf -creator “Foxit Reader”
fpdfmeta -i test -o output -creator “Foxit Reader”
fpdfmeta -i c:\input -o d:\output -creator “Foxit Reader”

fpdfmeta -i “c:\input*.pdf” -o d:\output -creator “Foxit Reader”

3.7.3.3 Open Password

- The optional argument (**-op**) indicates the open password for a password-protected input PDF file. It is not required if the input file is not password protected.

  Note *The output PDF file will retain the open password from the input file.*

Usage Example

1) Specify the open password for a password-protected input PDF file (-op 123)

   fpdfmeta -i c:\input\1.pdf -o d:\output\1_meta.pdf -author “Jessie” -subject “Foxit PDF Toolkit” -keywords “Foxit” -op 123
   fpdfmeta -i test\2.pdf -o output\2_meta.pdf -title “Foxit PDF Toolkit User Manual” -creator “Foxit Phantom” -op 123

2) Specify the open password for all input PDF files that have been protected with the same password (-op welcome)

   fpdfmeta -i c:\input -o d:\output -author “Jessie” -subject “Foxit PDF Toolkit” -keywords “Foxit” -op welcome
   fpdfmeta -i test -o output -title “Foxit PDF Toolkit User Manual” -creator “Foxit Phantom” -op welcome

   Note *It only supports typing one value for the argument (-op). Only files with the same open password can be processed together and files with different open password need to be processed separately.*

3.7.3.4 View metadata information

- The optional argument (**-properties**) is used to view metadata information of the PDF file designated by the **-i** option.

  Note *The argument (**-properties**) can be only used with **-i**, **-t**, **-op**, **-log** and **-l** options. The value of **-i** must be a PDF file, and the value of **-t** must be 1 if it is set.*

Usage Example

1) View the metadata information of the document (foxit.pdf) (-i c:\foxit.pdf -properties)

fpdfmeta -i c:\foxit.pdf -properties

2) View the metadata information of the document (user_manual.pdf) and specify the open password (-i c:\user_manual.pdf -op 123 -properties)

fpdfmeta -i c:\user_manual.pdf -op 123 -properties

3.7.3.5 Recursion Depth of Sub-folders

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard character like "c:\input*.pdf". By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.7.2 "Command Line Summary".

Usage Examples

1) Search the full folders (-r or -r 0)

fpdfmeta -i test -o output -title "Foxit PDF Toolkit User Manual" -author "Jessie" -r
fpdfmeta -i c:\input -o d:\output -subject "Foxit PDF Toolkit" -keywords "Foxit" -r
fpdfmeta -i c:\input*.pdf -o d:\output -creator "Foxit Reader"-r
fpdfmeta -i test -o output -title "Foxit PDF Toolkit User Manual" -author "Jessie" -r 0
fpdfmeta -i c:\input -o d:\output -subject "Foxit PDF Toolkit" -keywords "Foxit" -r 0
fpdfmeta -i "c:\input*.pdf" -o d:\output -creator "Foxit Reader" -r 0

2) Search only the current folder (-r 1)

fpdfmeta -i test -o output -title "Foxit PDF Toolkit User Manual" -author "Jessie" -r 1
fpdfmeta -i c:\input -o d:\output -subject "Foxit PDF Toolkit" -keywords "Foxit" -r 1
fpdfmeta -i "c:\input*.pdf" -o d:\output -creator "Foxit Reader" -r 1

Note *If you don't use this argument, the current folder will be searched by default. For example:*

fpdfmeta -i test -o output -title "Foxit PDF Toolkit User Manual" -author "Jessie"
fpdfmeta -i c:\input -o d:\output -subject "Foxit PDF Toolkit" -keywords "Foxit"
fpdfmeta -i "c:\input*.pdf" -o d:\output -creator "Foxit Reader"

3) Search the current folder and its sub-folders (-r 2)

fpdfmeta -i test -o output -title “Foxit PDF Toolkit User Manual” -author “Jessie” -r 2
fpdfmeta -i c:\input -o d:\output -subject “Foxit PDF Toolkit” -keywords “Foxit” -r 2
fpdfmeta -i “c:\input*.pdf” -o d:\output -creator “Foxit Reader” -r 2

3.7.3.6 Multi-thread Support

- The optional argument (**-t**) indicates the number of threads that are used to speed up batch programming by making full use of the CPU. By default, the number of the threads is 1.

  Note *It is recommended that you set the value of the number according to your computer’s CPU configuration.*

Usage Example

1) Set the number of threads to 3 (-t 3)

   fpdfmeta -i test -o output -title “Foxit PDF Toolkit User Manual” -author “Jessie” -t 3
   fpdfmeta -i c:\input -o d:\output -subject “Foxit PDF Toolkit” -keywords “Foxit” -t 3
   fpdfmeta -i “c:\input*.pdf” -o d:\output -creator “Foxit Reader” -t 3

3.7.3.7 Other Optional Arguments

a) Log file (-log<logfile> -l<log level>)

- The optional argument (**-log**) indicates the path of logfile, where the log message is placed. The argument (**-l**) indicates the log level. It is valid only when (**-log**) is used. By default, the log level is 4. For more details about this argument, please refer to section 3.7.2 “Command Line Summary”.

Usage Example

1) Save the log file to “d:\output\pdfmetadata.log” and set the log level to 3 (-log d:\output\pdfmetadata.log -l 3)

   fpdfmeta -i c:\input -o d:\output -keywords “Foxit” -log d:\output\pdfmetadata.log -l 3

b) Register information (-register <code> <licensee>)

- The optional argument (**-register**) is used to register the command line tool. The **<code>** is the activation code from Foxit and the **<licensee>** is the licensee name designated by the users.

Usage Example

7) Register the pdfmetadata tool with the code "77505-010G0-G1000-XMQ8D-2CR7R-TPBEI" and the licensee "Foxit" (-register 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foixt)

```
fpdfmeta -register 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foixt
```

c) License agreement (-license)

- The optional argument (**-license**) is used to print the license agreement.

Usage Example

1) Print the license information (-license)

```
fpdfmeta -license
```

d) Version information (-version/-v)

- The optional argument (-version/-v) is used to print the version information.

Usage Example

1) Print the version information (-version/-v)

```
fpdfmeta -version
fpdfmeta -v
```

e) Help information (-help/-h)

- The optional argument (**-help/-h**) is used to print the usage information.

Usage Example

1) Print the usage information (-help/-h)

```
fpdfmeta -help
fpdfmeta -h
```

3.8 PDF2Text

3.8.1 Basic Syntax

fpdf2txt *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> [-range <page range>] [-charcount] [-printcount] [-notype] [-nopagenumber] [-singlepage] [-encoding <text encoding>] [-op <password>] [-r [recursion]] [-t <thread>] [-log <logfile>] [-l <level>]*

fpdf2txt *-register <code> <licensee>*

fpdf2txt *-license*

fpdf2txt *-version/-v*

fpdf2txt *-help/-h*

Note:

- <> required
- [ ] optional
- / mutually exclusive
- A space is needed between the command line argument and the value

Only the <*-i <srcfile/srcfolder>*> and <*-o <destfile/destfolder>*> arguments are actually required. All others are optional, which are available for controlling the conversion as desired. The arguments could be given in any order. Full details on each are explained in the following section.

3.8.2 Command Line Summary

Note For some arguments whose values are strings, users can choose whether to add *quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i c:\input\1.pdf* *-i c:\input* *-i "c:\input*.pdf"*	Specifies the input file to be converted. ▪ The input string can be the name of a single PDF file or a folder. ▪ The file name can contain the wildcard character (*). For example, use *.pdf to include all PDF files in a given folder. **Note** Wildcard character (*.*) is currently not supported.

Option	Parameter	Description
-o	*<-o <string>>* *e.g.* *-o d:\output\1.txt* *-o d:\output*	Specifies the path of the output text file or folder. ▪ If the input is a single PDF file, the output should be a single text file, (e.g. *-o d:\output\1.txt*). ▪ If the input is a folder, the output should be a folder, (e.g. *-o d:\output*). **Note** The specified output path must already exist.
-range	*-rang <String>* *e.g.* *-range "1,5,9"* *-range "all"* *-range "even"* *-range "odd"* *-range "2-10,30"* *-range "10,50-"* *-range "odd,100-"*	Specifies a page range to convert. By default, all of the pages will be converted. ▪ Converts page 1,5, and 9: **-range "1,5,9"** ▪ Converts all pages: **-range "all"** ▪ Converts all even pages: **-range "even"** ▪ Converts all odd pages: **-range "odd"** ▪ Converts pages in the range from 2-10 and page 30: **-range "2-10,30"** ▪ Converts page 10 and all pages in the range from 50 to the last page: **-range "10,50-"** ▪ Converts all odd pages and all pages in the range from 100 to the last page: **-range "odd,100-"**
-charcount	*[-charcount]* *e.g.* *-charcount*	Gets the total number of characters on each page. **Note** The content of generated text files will only contain the number of characters on each page, if this argument is set.
-printcount	*[-printcount]* *e.g.* *-printcount*	Prints the number of characters to the screen. **Note:** The argument (**-printcount**) is valid only when (**-charcount**) is used.

Option	Parameter	Description
-notype	*[-notype]* *e.g.* *-notype*	Disables retaining the PDF page layout for the output text files.
-nopagenumber	*[-nopagenumber]* *e.g.* *-nopagenumber*	Disables retaining the page break for the output text files.
-singlepage	*<-singlepage>* *e.g.* *-singlepage*	Converts each PDF page into individual text files.
-encoding	*<-encoding <sting>>* *<-encoding <text encoding>>* *e.g.* *-encoding UTF8* *-encoding UTF16*	Specifies Unicode text encoding. The allowable value is UTF8 and UTF16. The default is UTF8.
-op	*[-op<string>]* *e.g.* *-op 123* *-op welcome*	Specifies the open password for the input file. Not required if the input file is not password protected.
-r	*[-r [integer]]* *e.g.* *-r* *-r 0* *-r 1* *-r 2* …	Specifies the number of layers to recurse when the input is a folder. • **-r 0 <-r>**: searches the full folders. • **-r 1**: searches only the current folder. • **-r 2**: searches the current folder and its sub-folders … **Note** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input PDF file or folder will be skipped if it is secured and the messages will be displayed.

Option	Parameter	Description
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2* *…*	Specifies the number of CPU threads to use. The default value is 1.
-log	*[-log <string>]* *e.g.* *-log d:\a.log*	Writes log information into a logfile at the specified existing path.
-l	*[-l <integer>]* *e.g.* *-l 1* *-l 2* *-l 3* *-l 4*	Sets the log level. The default is 4. • **-l 1**: logs messages only concerning program crashes. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1. • **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>* *e.g.* *-register* 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foxit	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.
-help/-h	*[-help]/[-h]* *e.g.* *-help* *-h*	Prints the usage information.
-version/-v	*[-version]/[-v]* *e.g.* *-version* *-v*	Prints the version information.

Option	Parameter	Description
-license	*[-license]* *e.g.* *-license*	Prints the license agreement.

3.8.3 Basic Usage

3.8.3.1 Input and Output

a) Input (-i)

- The input file should be a single PDF file or a folder. Users are not able to input multiple PDF files or folders, as well as a mixture composed of folders and PDF files. For example:

  -i c:\input\1.pdf (a single text file)
  -i c:\input (a single folder)

- It supports relative paths if the input file is in the current working folder. Users can input just the name of the PDF file or folder, instead of an absolute path. For example:

  -i test\1.pdf ("test\1.pdf" is in the current working folder)
  -i test ("test" folder is in the current working folder)

- It also supports wildcard characters, which are used to process multiple PDF files. For example:

  -i "c:\input*.pdf" (Only convert PDF files under "c:\input" folder)
  -i "test*.pdf" (Only convert PDF files under "test" folder)

  Note *When using wildcard characters in the input files, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

b) Output (-o)

- If the input is a single PDF file, you should specify the output path of a text file. If the input is a single folder, you should specify the output path of a folder. For example:

  -o d:\output\output.txt (a single text file)
  -o d:\output (a single folder)

  Note *The specified output path must already exist.*

- The output also supports relative paths if the specified output location is in the current working folder. Users can input just the name of the output text file or folder, instead of an absolute path. For example:

  -o output\output.txt ("output" folder is in the current working folder)
  -o output ("output" folder is in the current working folder)

Usage Examples

1) Convert a single PDF file to text file:

   fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt
   fpdf2txt -i test\2.pdf -o output\2.txt

2) Convert the PDF files in a folder to text files:

   fpdf2txt -i c:\input -o d:\output
   fpdf2txt -i test -o output
   fpdf2txt -i c:\input*.pdf -o d:\output
   fpdf2txt -i "test*.pdf" -o output

3.8.3.2 Conversion Settings

a) Page range to convert (-range)

- The optional argument (**-range**) is used to specify a page range to convert. If this argument is not set, all of the pages will be converted. For more details about this argument, please refer to section 3.8.2 "Command Line Summary".

Usage Example

1) Convert pages 2,3 and 8 (-range "2,3,8")

   fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -range "2,3,8"
   fpdf2txt -i test\2.pdf -o output\2.txt -range "2,3,8"
   fpdf2txt -i c:\input -o d:\output -range "2,3,8"
   fpdf2txt -i test -o output -range "2,3,8"
   fpdf2txt -i "c:\input*.pdf" -o d:\output -range "2,3,8"

2) Convert all even pages (-range "even")

fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -range “even”
fpdf2txt -i test\2.pdf -o output\2.txt -range “even”
fpdf2txt -i c:\input -o d:\output -range “even”
fpdf2txt -i test -o output -range “even”
fpdf2txt -i “c:\input*.pdf” -o d:\output -range “even”

3) Convert pages in the range from 2-10 and page 30 (-range “2-10,30”)

fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -range “2-10,30”
fpdf2txt -i test\2.pdf -o output\2.txt -range “2-10,30”
fpdf2txt -i c:\input -o d:\output -range “2-10,30”
fpdf2txt -i test -o output -range “2-10,30”
fpdf2txt -i “c:\input*.pdf” -o d:\output -range “2-10,30”

4) Convert all odd pages and all pages in the range from 100 to the last page (-range “odd,100-”)

fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -range “odd,100-”
fpdf2txt -i test\2.pdf -o output\2.txt -range “odd,100-”
fpdf2txt -i c:\input -o d:\output -range “odd,100-”
fpdf2txt -i test -o output -range “odd,100-”
fpdf2txt -i “c:\input*.pdf” -o d:\output -range “odd,100-”

b) Total number of characters on each page (-charcount)

- The optional argument (-charcount) is used to get the total number of characters on each page. The content of generated text files will only contain the number of characters on each page, if this argument is set.

Usage Example

1) Get the total number of characters on each page (-charcount)

fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -charcount
fpdf2txt -i test\2.pdf -o output\2.txt -charcount
fpdf2txt -i c:\input -o d:\output -charcount
fpdf2txt -i test -o output -charcount
fpdf2txt -i “c:\input*.pdf” -o d:\output -charcount

c) Print characters number to the screen (-printcount)

- The optional argument (-printcount) is used to print the number of characters to the screen. It is valid only when (**-charcount**) is used.

Usage Example

1) Print the number of characters to the screen (-printcount)

```
fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -charcount -printcount
fpdf2txt -i test\2.pdf -o output\2.txt -charcount -printcount
fpdf2txt -i c:\input -o d:\output -charcount -printcount
fpdf2txt -i test -o output -charcount -printcount
fpdf2txt -i “c:\input\*.pdf” -o d:\output -charcount -printcount
```

d) Ignore the PDF page layout (-notype)

- The optional argument (**-notype**) is used to ignore the PDF page layout. If this argument is set, the text file will not retain the PDF page layout.

Usage Example

1) Ignore the PDF page layout (-notype)

```
fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -notype
fpdf2txt -i test\2.pdf -o output\2.txt -notype
fpdf2txt -i c:\input -o d:\output -notype
fpdf2txt -i test -o output -notype
fpdf2txt -i “c:\input\*.pdf” -o d:\output -notype
```

e) Ignore the page break in the PDF (-nopagenumber)

- The optional argument (**-nopagenumber**) is used to ignore the page break in the PDF. If this argument is set, the text file will not retain the page break.

Usage Example

1) Ignore the page break in the PDF (-nopagenumber)

```
fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -nopagenumber
fpdf2txt -i test\2.pdf -o output\2.txt -nopagenumber
fpdf2txt -i c:\input -o d:\output -nopagenumber
```

fpdf2txt -i test -o output -nopagenumber
fpdf2txt -i "c:\input*.pdf" -o d:\output -nopagenumber

f) Convert each PDF page into individual text files (-singlepage)

- The optional argument (**-singlepage**) is used to convert each PDF page into individual text files.

Usage Example

1) Convert each PDF page into individual text files (-singlepage)

fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -singlepage
fpdf2txt -i test\2.pdf -o output\2.txt -singlepage
fpdf2txt -i c:\input -o d:\output -singlepage
fpdf2txt -i test -o output -singlepage
fpdf2txt -i "c:\input*.pdf" -o d:\output -singlepage

g) Text encoding (-encoding)

- The optional argument (**-encoding**) is used to specify Unicode text encoding. The allowable value is UTF8 and UTF16. By default, the text encoding is UTF8.

Usage Example

1) Set text encoding to UTF16 (-encoding UTF16)

fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -encoding UTF16
fpdf2txt -i test\2.pdf -o output\2.txt -encoding UTF16
fpdf2txt -i c:\input -o d:\output -encoding UTF16
fpdf2txt -i test -o output -encoding UTF16
fpdf2txt -i "c:\input*.pdf" -o d:\output -encoding UTF16

3.8.3.3 Open Password

- The optional argument (**-op**) indicates the open password for a password-protected input PDF file. It is not required if the input file is not password protected.

Usage Example

1) Specify the open password for a password-protected input PDF file (-op 123)

fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -op 123
fpdf2txt -i test\2.pdf -o output\2.txt -op 123
fpdf2txt -i c:\input -o d:\output -op 123
fpdf2txt -i test -o output -op 123
fpdf2txt -i "c:\input*.pdf" -o d:\output -op 123

2) Specify the open password for all input PDF files that have been protected with the same password (-op welcome)

fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -op welcome
fpdf2txt -i test\2.pdf -o output\2.txt -op welcome
fpdf2txt -i c:\input -o d:\output -op welcome
fpdf2txt -i test -o output -op welcome
fpdf2txt -i "c:\input*.pdf" -o d:\output -op welcome

Note *It only supports typing one value for the argument (**-op**). Only files with the same open password can be processed together and files with different open password need to be processed separately.*

3.8.3.4 Recursion Depth of Sub-folders

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard character like "c:\input*.pdf". By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.8.2 "Command Line Summary".

Usage Examples

1) Search the full folders (-r or -r 0)

fpdf2txt -i c:\input -o d:\output -r
fpdf2txt -i test -o output -r
fpdf2txt -i "c:\input*.pdf" -o d:\output -r
fpdf2txt -i c:\input -o d:\output -r 0
fpdf2txt -i test -o output -r 0
fpdf2txt -i "c:\input*.pdf" -o d:\output -r 0

2) Search only the current folder (-r 1)

fpdf2txt -i c:\input -o d:\output -r 1
fpdf2txt -i test -o output -r 1
fpdf2txt -i "c:\input*.pdf" -o d:\output -r 1

Note *If you don't use this argument, the current folder will be searched by default. For example:*

fpdf2txt -i c:\input -o d:\output
fpdf2txt -i test -o output
fpdf2txt -i "c:\input*.pdf" -o d:\output

3) Search the current folder and its sub-folders (-r 2)

fpdf2txt -i c:\input -o d:\output -r 2
fpdf2txt -i test -o output -r 2
fpdf2txt -i "c:\input*.pdf" -o d:\output -r 2

3.8.3.5 Multi-thread Support

- The optional argument (**-t**) indicates the number of threads that are used to speed up batch programming by making full use of the CPU. By default, the number of threads is 1.

  Note *It is recommended that you set the value of the number according to your computer's CPU configuration.*

Usage Example

1) Set the number of threads to 3 (-t 3)

fpdf2txt -i c:\input -o d:\output -t 3
fpdf2txt -i test -o output -t 3
fpdf2txt -i "c:\input*.pdf" -o d:\output -t 3

3.8.3.6 Other Optional Arguments

a) Log file (-log<logfile> -l<log level>)

- The optional argument (**-log**) indicates the path of logfile, where the log message is placed. The argument (**-l**) indicates the log level. It is valid only when (**-log**) is used. By default, the log level is 4. For more details about this argument, please refer to section 3.8.2 "Command Line Summary".

Usage Example

1) Save the log file to "d:\output\pdf2text.log" and set the log level to 3 (-log d:\output\pdf2text.log -l 3)

```
fpdf2txt -i c:\input -o d:\output -log d:\output\pdf2text.log -l 3
```

b) Register information (-register <code> <licensee>)

- The optional argument (**-register**) is used to register the command line tool. The **<code>** is the activation code from Foxit and the **<licensee>** is the licensee name designated by the users.

Usage Example

8) Register the pdf2text tool with the code "77505-010G0-G1000-XMQ8D-2CR7R-TPBEI" and the licensee "Foxit" (-register 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foixt)

```
fpdf2txt -register 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foixt
```

c) License agreement (-license)

- The optional argument (**-license**) is used to print the license agreement.

Usage Example

1) Print the license agreement (-license)

```
fpdf2txt -license
```

d) Version information (-version/-v)

- The optional argument (-version/-v) is used to print the version information.

Usage Example

1) Print the version information (-version/-v)

```
fpdf2txt -version
fpdf2txt -v
```

e) Help information (-help/-h)

- The optional argument (**-help/-h**) is used to print the usage information.

Usage Example

1) Print the usage information (-help/-h)

```
fpdf2txt -help
fpdf2txt -h
```

3.9 Text2PDF

3.9.1 Basic Syntax

ftxt2pdf *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> [-width <PDF width>] [-height <PDF height>]*
[-margin <left [top right bottom]>] [-font <fontname>] [-fs <fontsize>] [-fontcolor <R> <G> <B>]
[-sp <password>] [-title <title>] [-subject <subject>] [-keywords <keywords>] [-author <author>]
[-creator <creator>] [-r [recursion]] [-t <thread>] [-log <logfile>] [-l <level>]

ftxt2pdf *-register <code> <licensee>*

ftxt2pdf *-license*

ftxt2pdf *-version/-v*

ftxt2pdf *-help/-h*

Note:

- <> required
- [ ] optional
- / mutually exclusive
- A space is needed between the command line argument and the value

Only the <*-i <srcfile/srcfolder>*> and <*-o <destfile/destfolder>*> arguments are actually required. All others are optional, which are available for controlling the conversion as desired. The arguments could be given in any order. Full details on each are explained in the following section.

3.9.2 Command Line Summary

Note For some arguments whose values are strings, users can choose whether to add *quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i c:\input\1.txt* *-i c:\input* *-i "c:\input*.txt"*	Specifies the input file to be converted. ▪ The input string can be the name of a single text file or a folder. ▪ The file name can contain the wildcard character (*). For example, use *.txt to include all text files in a given folder. **Note** Wildcard character (*.*) is currently not supported.

Option	Parameter	Description
-o	*<-o <string>>* *e.g.* *-o d:\output\1.pdf* *-o d:\output*	Specifies the path of the output PDF file or folder. ▪ If the input is a single text file, the output should be a single PDF file, (e.g. *-o d:\output\1.pdf*). ▪ If the input is a folder, the output should be a folder, (e.g. *-o d:\output*). **Note** The specified output path must already exist.
-width	*[-width <points>]* *e.g.* *-width 612*	Sets the page width of the output PDF file in points. Default width value: 595 points. Allowable range: 8-14400 points.
-height	*[-height <points>]* *e.g.* *-height 792*	Sets the page height of the output PDF file in points. Default height value: 842 points. Allowable range: 8-14400 points.
-margin	*[-margin <points [points points points]>]* *-margin <left [top right bottom]>* *e.g.* *-margin 20* *-margin 10 20* *-margin 10 20 0* *-margin 10 20 0 40*	Sets size of margin for each PDF page in points. Default margin values: 60 72 60 72. Allowable values: 0-size of page in points; in addition, the sum of the left and right values must be less than the width of the page, and the sum of the top and bottom values must be less than the height of the page. ***-margin left top right bottom*** **-margin 20**: sets the left margin to 20 points. **-margin 10 20**: sets the left margin to 10 points and the top margin to 20 points. **-margin 10 20 0**: sets the left margin to 10 points, the top margin to 20 points, and the right margin to 0 points. **-margin 10 20 0 40**: sets the left margin to 10 points, the top margin to 20 points, the right margin to 0 points, and the bottom margin to 40 points.

Option	Parameter	Description
-font	*[-font <string>]* *-font <fontname>* *e.g.* *-font Calibri* *-font Helvetica* *-font Arial* *…*	Sets the font style of the text in PDF files to be converted. **Note** The font style should be installed in a local environment, otherwise the default font style will be used.
-fs	*[-fs <integer>]* *-fs <fontsize>* *e.g.* *-fs 8* *-fs 11* *-fs 72*	Sets the font size of the text in PDF files to be converted. Default value: 9. Allowable range: 8-72.
-fontcolor	*[-fontcolor <integer> <integer> <integer>]* *-fontcolor <R> <G> <B>* *e.g.* *-fontcolor 0 0 0* *-fontcolor 255 255 0* *-fontcolor 255 255 255* *…*	Sets the font color of the text in PDF files to be converted. By default, the font color of the output PDF is black. Allowable range of the values for each RGB component is 0-255.
-sp	*[-sp<string>]* *e.g.* *-sp welcome*	Sets the document open password of the output PDF as the "string". By default, there is no password.
-title	*<-title <string>>* *e.g.* *-title "Foxit PDF Toolkit User Manual"*	Sets title of PDF files.
-subject	*<-subject <string>>* *e.g.* *-subject "Foxit PDF Toolkit"*	Sets subject of PDF files.
-keywords	*[-keywords <string>]* *e.g.* *-keywords "Foxit"*	Sets keywords of PDF files.

Option	Parameter	Description
-author	*[-author <string>]* *e.g.* *-author "Jessie"*	Sets author of PDF files.
-creator	*[-creator <string>]* *e.g.* *-creator "Foxit PhantomPDF"* *-creator "Foxit Reader"* *-creator "Microsoft® Word 2013"*	Sets creator of PDF files. **Note** It indicates the name of the application that created the source document from which a PDF is generated. For example, if the source document was created from Microsoft® *Word 2013, then you can set the creator to "*Microsoft® *Word 2013".*
-r	*[-r [integer]]* *e.g.* *-r* *-r 0* *-r 1* *-r 2* …	Specifies the number of layers to recurse when the input is a folder. • **-r 0 <-r>**: searches the full folders. • **-r 1**: searches only the current folder. • **-r 2**: searches the current folder and its sub-folders … **Note** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input folder will be skipped if it is secured and the messages will be displayed.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2* …	Specifies the number of CPU threads to use. The default value is 1.
-log	*[-log <string>]* *e.g.* *-log d:\a.log*	Writes log information into a logfile at the specified existing path.

Option	Parameter	Description
-l	*[-l <integer>]* *e.g.* *-l 1* *-l 2* *-l 3* *-l 4*	Sets the log level. The default is 4. • **-l 1**: logs messages only concerning program crashes. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1. • **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>* *e.g.* *-register* 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foxit	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.
-help/-h	*[-help]/[-h]* *e.g.* *-help* *-h*	Prints the usage information.
-version/-v	*[-version]/[-v]* *e.g.* *-version* *-v*	Prints the version information.
-license	*[-license]* *e.g.* *-license*	Prints the license agreement.

3.9.3 Basic Usage

3.9.3.1 Input and Output

a) Input (-i)

- The input file should be a single text file or a folder. Users are not able to input multiple files or folders, as well as a mixture composed of folders and text files. For example:

  -i c:\input\1.txt (a single text file)
  -i c:\input (a single folder)

- It supports relative paths if the input file is in the current working folder. Users can input just the name of the PDF file or folder, instead of an absolute path. For example:

  -i test\1.txt ("test\1.txt" is in the current working folder)
  -i test ("test" folder is in the current working folder)

- It also supports wildcard characters, which are used to process multiple files. For example:

  -i "c:\input*.txt" (Only convert text files under "c:\input" folder)
  -i "test*.txt" (Only convert text files under "test" folder)

  Note *When using wildcard characters in the input files, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

b) Output (-o)

- If the input is a single text file, you should specify the output path of a PDF file. If the input is a single folder, you should specify the output path of a folder. For example:

  -o d:\output\output.pdf (a single PDF file)
  -o d:\output (a single folder)

  Note *The specified output path must already exist.*

- The output also supports relative paths if the specified output location is in the current working folder. Users can input just the name of the output PDF file or folder, instead of an absolute path. For example:

  -o output\output.pdf ("output" folder is in the current working folder)
  -o output ("output" folder is in the current working folder)

Usage Examples

1) Convert a single text file to PDF:

ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf
ftxt2pdf -i test\2.txt -o output\2.pdf

2) Convert the text files in a folder to PDF:

ftxt2pdf -i c:\input -o d:\output
ftxt2pdf -i test -o output
ftxt2pdf -i c:\input*.txt -o d:\output
ftxt2pdf -i test*.txt -o output

3.9.3.2 PDF/page Settings for Conversion

a) Page size setting (-width, -height)

- The optional arguments (**-width**) and (**-height**) are used to set the page width and height for the output PDF file in points. The default width and height are 595 and 842 points respectively with the allowable range of 8-14400 points.

Usage Example

1) Set the page width and height to 400 and 300 for the output PDF file (-width 400 -height 300)

ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -width 400 -height 300
ftxt2pdf -i c:\input -o d:\output -width 400 -height 300
ftxt2pdf -i test -o output -width 400 -height 300
ftxt2pdf -i "c:\input*.txt" -o d:\output -width 400 -height 300

b) Margin setting (-margin)

- The optional argument (-margin) is used to set size of margin for each PDF page in points. The default margin values are 60 72 60 72. For more details about this argument, please refer to section 3.9.2 "Command Line Summary".

Note *The sum of the left and right values must be less than the width of the page, and the sum of the top and bottom values must be less than the height of the page.*

Usage Example

1) Set the left margin to 20 points (-margin 20)

ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -margin 20
ftxt2pdf -i c:\input -o d:\output -margin 20
ftxt2pdf -i test -o output -margin 20
ftxt2pdf -i "c:\input*.txt" -o d:\output -margin 20

2) Set the left margin to 20 points, and the top margin to 10 points (-margin 20 10)

ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -margin 20 10
ftxt2pdf -i c:\input -o d:\output -margin 20 10
ftxt2pdf -i test -o output -margin 20 10
ftxt2pdf -i "c:\input*.txt" -o d:\output -margin 20 10

3) Set the left margin to 10 points, the top margin to 10 points and the right margin to 30 points (-margin 10 10 30)

ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -margin 10 10 30
ftxt2pdf -i c:\input -o d:\output -margin 10 10 30
ftxt2pdf -i test -o output -margin 10 10 30
ftxt2pdf -i "c:\input*.txt" -o d:\output -margin 10 10 30

4) Set the left margin to 10 points, the top margin to 10 points, the right margin to 30 points and the bottom margin to 20 points (-margin 10 10 30 20)

ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -margin 10 10 30 20
ftxt2pdf -i c:\input -o d:\output -margin 10 10 30 20
ftxt2pdf -i test -o output -margin 10 10 30 20
ftxt2pdf -i "c:\input*.txt" -o d:\output -margin 10 10 30 20

c) Password setting (-sp)

- The optional argument (**-sp**) is used to set the open password for the output PDF file. By default, the output PDF file has no open password.

Usage Example

1) Set the open password to "welcome" for the output PDF files (-sp welcome)

ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -sp welcome
ftxt2pdf -i c:\input -o d:\output -sp welcome
ftxt2pdf -i test -o output -sp welcome

ftxt2pdf -i “c:\input*.txt” -o d:\output -sp welcome

3.9.3.3 Font Settings

a) Text font (-font)

- The optional arguments (**-font**) is used to set the font style of the text in PDF files to be converted.

  Note *The font style should be installed in a local environment, otherwise the default font style will be used.*

Usage Example

1) Set the font style to “Calibri” (-font “Calibri”)

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -font “Calibri”
ftxt2pdf -i c:\input -o d:\output -font “Calibri”
ftxt2pdf -i test -o output -font “Calibri”
ftxt2pdf -i “c:\input\*.txt” -o d:\output -font “Calibri”
```

b) Text font size (-fs)

- The optional argument (**-fs**) is used to set the font size of the text in PDF files to be converted. The default value is set to 9, and the allowable range is from 8 to 72.

Usage Example

1) Set the font size to 12 (-fs 12)

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -fs 12
ftxt2pdf -i c:\input -o d:\output -fs 12
ftxt2pdf -i test -o output -fs 12
ftxt2pdf -i “c:\input\*.txt” -o d:\output -fs 12
```

c) Text font color (-fontcolor)

- The optional argument (**-fontcolor**) is used to set the font color of the text in PDF files to be converted. By default, the font color is black. The allowable range of the values for each RGB component is from 0 to 255.

Usage Example

1) Set the font color to blue (-fontcolor 0 0 255)

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -fontcolor 0 0 255
ftxt2pdf -i c:\input -o d:\output -fontcolor 0 0 255
ftxt2pdf -i test -o output -fontcolor 0 0 255
ftxt2pdf -i "c:\input\*.txt" -o d:\output -fontcolor 0 0 255
```

3.9.3.4 Document Metadata Settings

a) Title (-title)

- The optional argument (**-title**) is used to set title of PDF files.

Usage Example

1) Set document title to "Foxit PDF Toolkit User Manual" (-title "Foxit PDF Toolkit User Manual")

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -title "Foxit PDF Toolkit User Manual"
ftxt2pdf -i c:\input -o d:\output -title "Foxit PDF Toolkit User Manual"
ftxt2pdf -i test -o output -title "Foxit PDF Toolkit User Manual"
ftxt2pdf -i "c:\input\*.txt" -o d:\output -title "Foxit PDF Toolkit User Manual"
```

b) Subject (-subject)

- The optional argument (**-subject**) is used to set subject of PDF files.

Usage Example

1) Set document subject to "Foxit PDF Toolkit" (-subject "Foxit PDF Toolkit")

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -subject "Foxit PDF Toolkit"
ftxt2pdf -i c:\input -o d:\output -subject "Foxit PDF Toolkit"
ftxt2pdf -i test -o output -subject "Foxit PDF Toolkit"
ftxt2pdf -i "c:\input\*.txt" -o d:\output -subject "Foxit PDF Toolkit"
```

c) Keywords (-keywords)

- The optional argument (**-keywords**) is used to set keywords of PDF files.

Usage Example

1) Set document keywords to “Foxit” (-keywords “Foxit”)

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -keywords “Foxit”
ftxt2pdf -i c:\input -o d:\output -keywords “Foxit”
ftxt2pdf -i test -o output -keywords “Foxit”
ftxt2pdf -i “c:\input\*.txt” -o d:\output -keywords “Foxit”
```

d) Author (-author)

- The optional argument (**-author**) is used to set an author of PDF files.

Usage Example

1) Set document author to “Jessie” (-author “Jessie”)

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -author “Jessie”
ftxt2pdf -i c:\input -o d:\output -author “Jessie”
ftxt2pdf -i test -o output -author “Jessie”
ftxt2pdf -i “c:\input\*.txt” -o d:\output -author “Jessie”
```

e) Creator (-creator)

- The optional argument (**-creator**) is used to set file creation application information of PDF files.

Usage Example

1) Set document creator to “Foxit Reader” (-creator “Foxit Reader”)

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -creator “Foxit Reader”
ftxt2pdf -i c:\input -o d:\output -creator “Foxit Reader”
ftxt2pdf -i test -o output -creator “Foxit Reader”
ftxt2pdf -i “c:\input\*.txt” -o d:\output -creator “Foxit Reader”
```

3.9.3.5 Recursion Depth of Sub-folders

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard character like “c:\input*.txt”. By default, the

recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to 3.9.2 "Command Line Summary".

Usage Examples

1) Search the full folders (-r or -r 0)

```
ftxt2pdf -i c:\input -o d:\output -r
ftxt2pdf -i test -o output -r
ftxt2pdf -i "c:\input\*.txt" -o d:\output -r
ftxt2pdf -i c:\input -o d:\output -r 0
ftxt2pdf -i test -o output -r 0
ftxt2pdf -i "c:\input\*.txt" -o d:\output -r 0
```

2) Search only the current folder (-r 1)

```
ftxt2pdf -i c:\input -o d:\output -r 1
ftxt2pdf -i test -o output -r 1
ftxt2pdf -i "c:\input\*.txt" -o d:\output -r 1
```

Note *If you don't use this argument, the current folder will be searched by default. For example:*

```
ftxt2pdf -i c:\input -o d:\output
ftxt2pdf -i test -o output
ftxt2pdf -i "c:\input\*.txt" -o d:\output
```

3) Search the current folder and its sub-folders (-r 2)

```
ftxt2pdf -i c:\input -o d:\output -r 2
ftxt2pdf -i test -o output -r 2
ftxt2pdf -i "c:\input\*.txt" -o d:\output -r 2
```

3.9.3.6 Multi-thread Support

- The optional argument (**-t**) indicates the number of threads that are used to speed up batch programming by making full use of the CPU. By default, the number of threads is 1.

  Note *It is recommended that you set the value of the number according to your computer's CPU configuration.*

Usage Example

1) Set the number of threads to 3 (-t 3)

```
ftxt2pdf -i c:\input -o d:\output -t 3
ftxt2pdf -i test -o output -t 3
ftxt2pdf -i "c:\input\*.txt" -o d:\output -t 3
```

3.9.3.7 Other Optional Arguments

a) Log file (-log<logfile> -l<log level>)

- The optional argument (**-log**) indicates the path of logfile, where the log message is placed. The argument (**-l**) indicates the log level. It is valid only when (**-log**) is used. By default, the log level is 4. For more details about this argument, please refer to section 3.9.2 "Command Line Summary".

Usage Example

1) Save the log file to "d:\output\text2pdf.log" and set the log level to 3 (-log d:\output\text2pdf.log -l 3)

```
ftxt2pdf -i c:\input -o d:\output -log d:\output\text2pdf.log -l 3
```

b) Register information (-register <code> <licensee>)

- The optional argument (**-register**) is used to register the command line tool. The **<code>** is the activation code from Foxit and the **<licensee>** is the licensee name designated by the users.

Usage Example

1) Register the text2pdf tool with the code "77505-010G0-G1000-XMQ8D-2CR7R-TPBEI" and the licensee "Foxit" (-register 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foixt)

```
ftxt2pdf -register 77505-010G0-G1000-XMQ8D-2CR7R-TPBEI Foixt
```

c) License agreement (-license)

- The optional argument (**-license**) is used to print the license agreement.

Usage Example

1) Print the license agreement (-license)

ftxt2pdf -license

d) Version information (-version/-v)

- The optional argument (-version/-v) is used to print the version information.

Usage Example

1) Print the version information (-version/-v)

ftxt2pdf -version
ftxt2pdf -v

e) Help information (-help/-h)

- The optional argument (**-help/-h**) is used to print the usage information.

Usage Example

1) Print the usage information (-help/-h)

ftxt2pdf -help
ftxt2pdf -h

3.10RMS

3.10.1Basic Syntax

RMSProtector *[/decrypt <location>]*
[/encrypt <location> </template <name> [issuer]> [/highstrength] [/revoke]
[/MicrosoftIRMV1]]
[/encrypt <location> </user <name> /rights <rights> [issuer]> [/highstrength] [/revoke]
[/MicrosoftIRMV1]]
[/showtemplates [/sync]] [/preserveattributes]
[/showencryption <location>]
[/log <log_file> [/append] [/simple]] [/silent]
RMSProtector */register <code> <licensee>*
RMSProtector */license*

Note:

- <> required
- [ ] optional
- A space is needed between the command line argument and the value

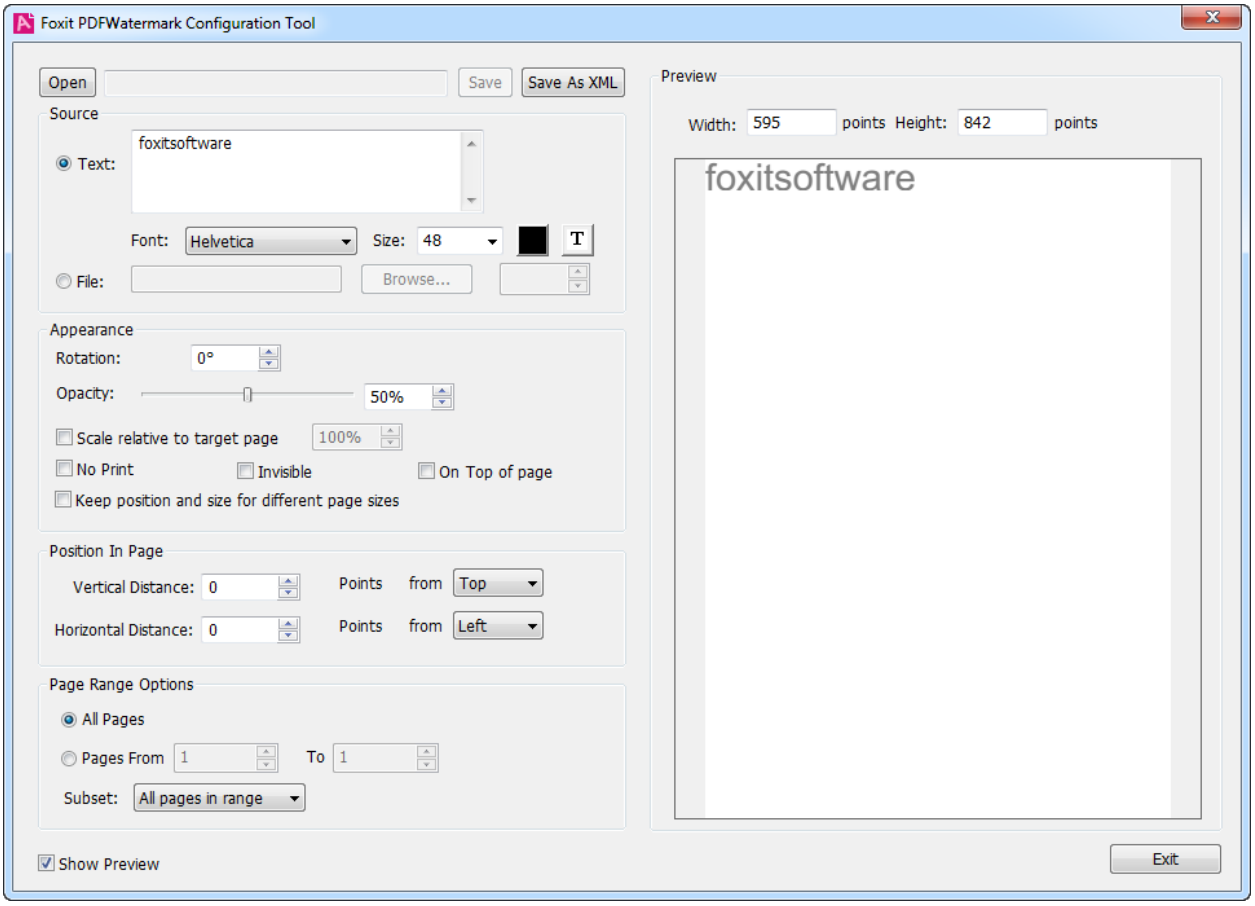
For the RMS tool, please go to the "rms" folder in the installation package for a more detailed introduction.

4 Foxit Configuration Tool

4.1 Watermark Configuration Tool

Foxit PDFWatermark Configuration Tool is used to set the properties of watermarks and then save them as a configuration file with the extension ".xml", which is required in the command line. The watermark settings contain four main parts: source, appearance, position in page, and page range options. We will give a detailed introduction on each part in the following sections.

First, go to "configtool" folder, double-click the fpdfwmconf.exe or fpdfwmconf64.exe to launch the Foxit PDFWatermark Configuration Tool, and then you can see the configuration screen as follows.



PDFWatermark Configuration Screen

Users can click the "**Open**" button to load an existing watermark configuration file (.xml), and then edit its settings according to their desired requirement. After editing, users can click "**Save**" button to save the changes or click "**Save As XML**" button to save it as a new watermark configuration file.

Check "**Show Preview**" to preview how the text/image/PDF watermark will be displayed in the PDF file. By default, it is checked.

4.1.1 Watermark Settings

4.1.1.1 Source

Foxit PDFWatermark supports three types of watermarks: text, image and PDF.

Text Watermark

- **Text**

  Users can enter text in the content box, which will be appeared as text watermark in the preview window.

- **Font**

  This option is activated only when Text Watermark is selected. Users can choose the font name, font size, and font color from the drop-down menus.

- **Underline**

  Users can click  icon to set underline for text when Text Watermark is selected.

Image Watermark

- **File**

  Users can browse their computer and choose an image that will be appeared as image watermark in preview window. The following image formats are currently supported: BMP, DIB, JPG, JPEG, JPE, GIF, PNG, TIFF and TIF.

PDF Watermark

- **File**

Users can browse their computer and choose a PDF file. The preview window will display the first page of the PDF file, you can click the selection box on the right side of the "**Browse…**" button to preview other pages and decide which PDF page will be used to set as PDF watermark.

4.1.1.2 Appearance

This section introduces how to set the appearance for text/image/PDF watermarks.

- **Rotation**

  Users can enter an angle to rotate the text/image/PDF watermark. The rotation angle can be set between 0 and 360.

- **Opacity**

  This option is used to set the transparence for text/image/PDF watermarks. The value is from 0 to 100 (0 meaning invisible, and 100 meaning visibly solid).

- **Scale relative to target page**

  Check "**Scale relative to target page**" and enter a number in the percentage box to resize the text/image/PDF watermark in relation to the PDF page's dimensions.

  Note *The drop-down box of font size will be empty if you check "**Scale relative to target page**". And if you reset the font size, the "**Scale relative to target page**" will be unchecked.*

- **No Print**

  This option is used to control the visibility for text/image/PDF watermarks when you print the PDF. If you check "**No Print**", the added watermark will not be shown when printing. By default, it is not checked.

- **Invisible**

  This option is used to control the visibility for text/image/PDF watermarks on screen. If you check "**Invisible**", the added watermark will not be shown on screen. By default, it is not checked.

- **On Top of page**

If you Check "**On Top of page**", the text/image/PDF watermark will be placed on top of the page content, that is, the watermark may cover some content. Otherwise, the watermark will be placed under the content, and the page content may obstruct your view of some part of the watermark. By default, it is not checked.

- **Keep position and size for different page size**

  This option is used to control watermark variations in a PDF with pages of varying sizes. If you check "**Keep position and size for different page size**", the added watermark will be the same in different pages.

  Note *Please do not check "**Keep position and size for different page size**" and "**Scale relative to target page**" simultaneously, or the "**Keep position and size for different page size**" will be omitted, that is, it will not have any effect.*

4.1.1.3 Position in page

Watermark position in page is controlled by **Vertical Distance** and **Horizontal Distance** positioning. The distance unit is points. Users can set the relative benchmark for the distance, such as Top, Bottom, Left, Right and Center.

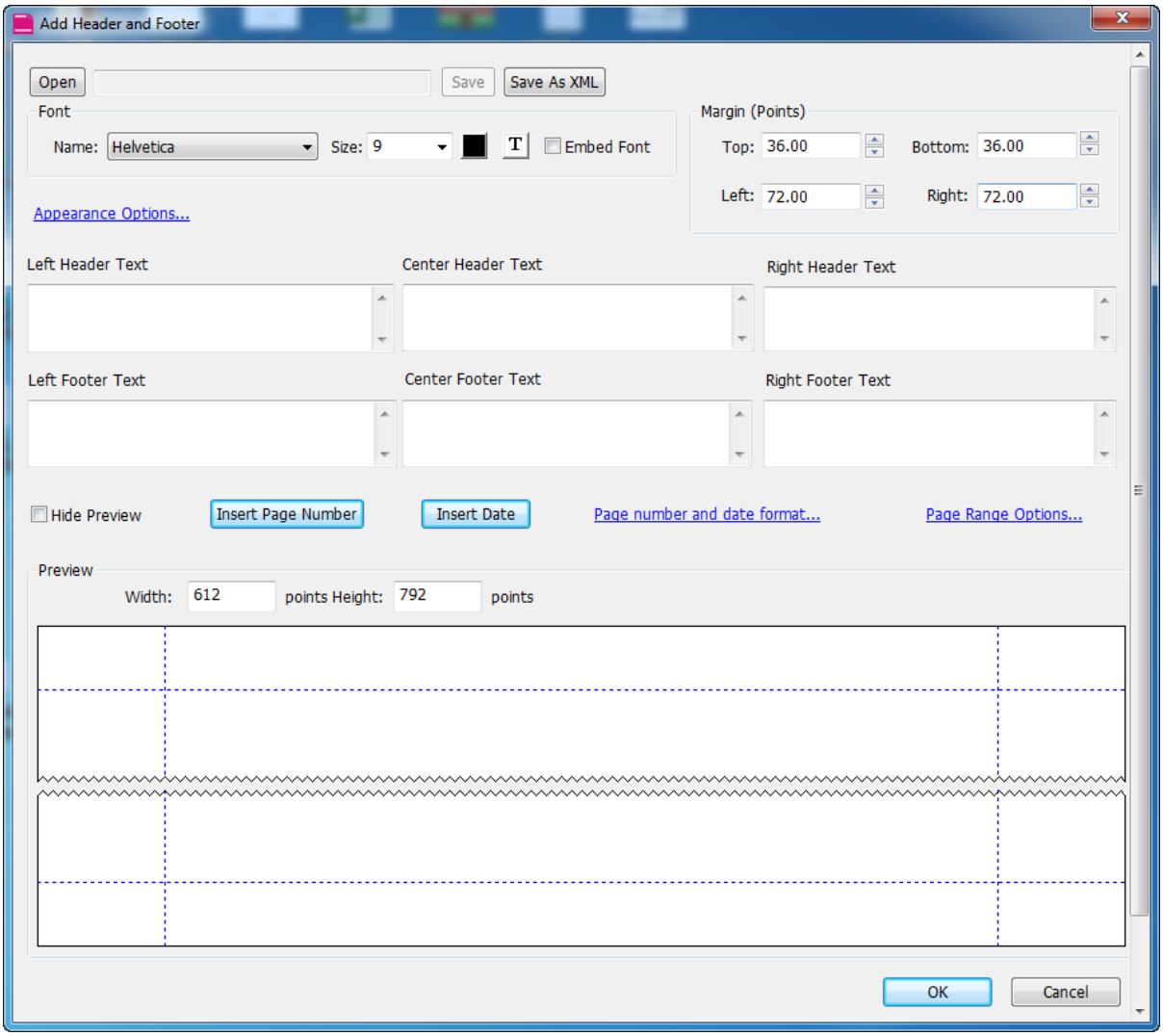
4.1.1.4 Page Range Options

Page range options are used to choose the page range to add watermark. Users can select all pages or specify the page range, or choose even or odd pages via clicking the right items in the subset list.

4.2 PDFHeaderFooter Configuration Tool

Foxit PDFHeaderFooter Configuration Tool is used to set the properties of header/footer and then save them as a configuration file with extension ".xml" which is required in the command line. The header/footer settings contain six main parts: font, margin, appearance options, text, page number and date format, and page range options. We will give a detailed introduction on each part in the following sections.

First, go to "configtool" folder, double-click the fpdfhfconf.exe or fpdfhfconf64.exe to launch Foxit PDFHeaderFooter Configuration Tool, and then you can see the configuration screen as follows.



PDFHeaderFooter Configuration Screen

Users can click the "**Open**" button to load an existing header/footer configuration file (.xml), and then edit its settings according to their desired requirement. After editing, users can click "**Save**" button to save the changes or click "**Save As XML**" button to save it as a new header/footer configuration file.

Check "**Hide Preview**" to hide the preview window about how the header/footer will be displayed in the PDF file. By default, it is unchecked.

4.2.1 Header/Footer Settings

4.2.1.1 Font

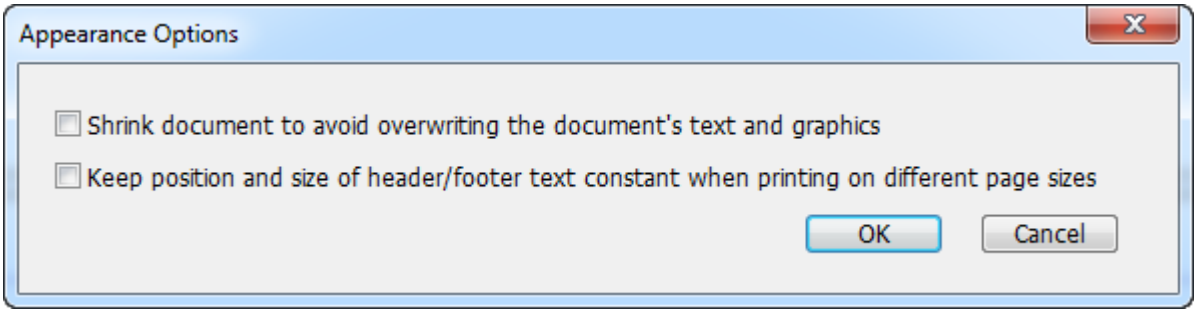
The font option allows users to choose the font name, font size and font color from the drop-down menus, to click the T icon to set underline for text, and to check the "**Embed Font"** to set the text font as embedded font.

4.2.1.2 Margin

The margin option is used to control the header/footer position in PDF page. The unit is "points". Users can set margins in four directions for the added header/footer, including top, bottom, left and right.

4.2.1.3 Appearance Options

There are two options you can choose when you click on "Appearance Options" as shown in the following figure. One is "**Shrink document to avoid overwriting the document's text and graphics**", which can be used to shrink the document when the added header/footer may overwrite the text or graphic in the document. The other is "**Keep position and size of header/footer text constant when printing on different page sizes**", which can keep the added header/footer at the same position in different pages.



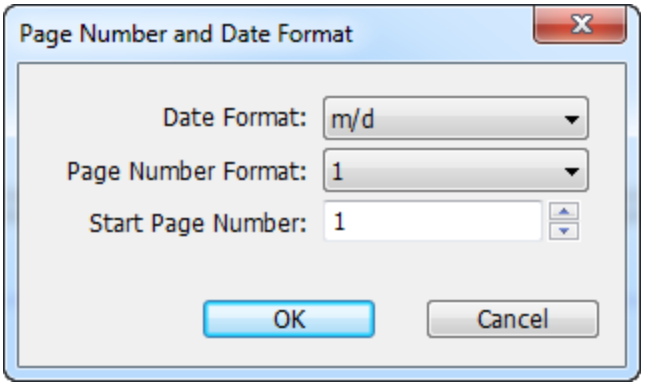
Appearance Options

4.2.1.4 Text

Six text-input boxes are provided to input the texts of header/footer you want to add. It includes left header text, center header text, right header text, left footer text, center footer text and right footer text. Users can input text to the desired box to add the header/footer.

4.2.1.5 Page Number and Date Format

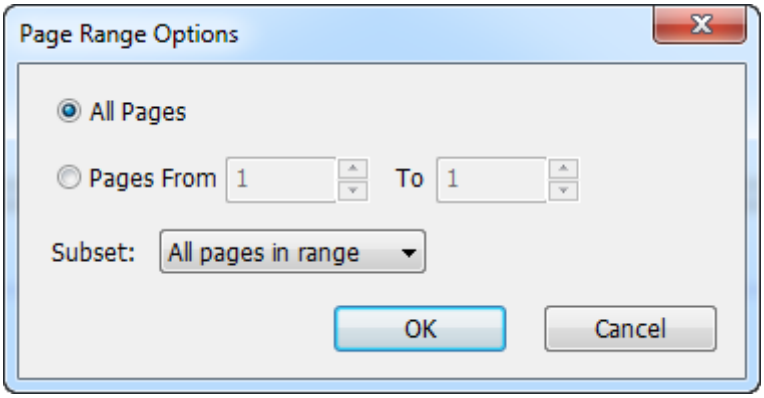
Click "**Page Number and Date Format**" to open the setting window as shown in the following figure. Users can set the Date Format, Page Number Format and Start Page Number. After setting, click the "**Insert Page Number**" button to insert page number to the desired text-input box, and click the "**Insert Date**" button to insert date to the desired text-input box.



Page Number and Date Format

4.2.1.6 Page Range Options

Click the "**Page Range Options**" to open the settings window as shown in the following figure. Page range options are used to choose the page range to add header/footer. Users can select all pages or specify the page range, or choose even pages or odd pages via clicking the right items in the subset list.



Page Range Options

5 Working with API

Foxit PDF Toolkit provides another way for users who want to perform PDF manipulation through API.

Note *To integrate the Foxit PDF Toolkit into your own applications with API, please contact Foxit sales team to purchase Enterprise License.*

Each module offers two types of simple-to-use APIs including **char** and **wchar_t** API. Users can choose one of them to call according to the requirements of the applications. In this manual, we take char API as an example to show how to integrate the Foxit PDF Toolkit into your own applications with API. Four functions are required.

First, initialize Foxit PDF Toolkit library and check the license.

```
int FXT_InitLibrary(const char* key, int screenFlag);
```

The parameter "**key**" is the path of the license file ("**ftlkey.txt**", generated in the installed path after registering the tool using the activation code purchased from Foxit.).The parameter "**screenFlag**" controls whether to print the output information to screen. If the value is 1, print the output information to screen, not vice versa.

Second, call the function of the desired tool.

```
int FXT_Image2PDFRun(const char* commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_Office2PDFRun(const char* commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_WatermarkRun(const char* commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_HeaderFooterRun(const char* commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_OptimizerRun(const char* commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_RedactorRun(const char* commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_MetadataRun(const char* commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_PDF2TextRun(const char* commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_Text2PDFRun(const char* commandline, FXT_CallbackFun callback,  void* userData = 0);
```

Select the corresponding function of the tool. Where, the parameter "**commandline**" is a command string which is exactly the same as the general syntax used for the command line application (e.g. "-i c:\input -o d:\output"). The parameter "**callback**" is the callback function provided for users to do some special processing. The parameter "**userData**" is a pointer used to transfer user data.

Third, declare the callback function.

```
typedef char*(*FXT_CallbackFun)(void* userData, int mode, char* msg, bool* isStop)
```

The parameter "**useData**" is a pointer of user data. The parameter "**mode**" specifies the output mode of information including CALLBACK_PDFTOOL_RUN_ERROR, CALLBACK_PDFTOOL_PARAM_ERROR, CALLBACK_PDFTOOL_MSG and CALLBACK_PDFTOOL_PASSWORD. The parameter "**msg**" is the output message when calling the callback function. The parameter "**isStop**" controls whether to stop the current application. If the value is true, stop the current application, otherwise, continue running the application.

Last, release and destroy Foxit PDF Toolkit library.

```
void FXT_DestroyLibrary();
```

For more details about the API, please refer to the header file "fxpdftools.h" in the "include" folder in the installation path.

The following are some examples on how to work with API for each tool.

5.1 Image2PDF

5.1.1 Working with Image2PDF API

The following is the simplest application that can be built using Image2PDF API:

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 0);
	FXT_Image2PDFRun("-i d:\\input -o d:\\output\\output.pdf", myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

This application converts all the image files in the "d:\input" folder into one PDF file (output.pdf). To convert each image file into individual PDF files, just delete "output.pdf" from the command string.

The command string (`"-i d:\\input -o d:\\output\\output.pdf"`) of the above application is set by users directly in advance. Users also can get the command string through the command line. Building a command line application using Image2PDF API is as simple as the following:

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 0);

	std::string strCommandline = "";
	for(int i = 1; i < argc; i++)
		strCommandline.append(argv[i]).append(" ");
	FXT_Image2PDFRun(strCommandline.c_str(), myCallBack, NULL);
```

```
	FXT_DestroyLibrary();
}
```

It is also possible to build the command string dynamically (e.g. based on the user or a dynamic input) as illustrated in the following code snippet:

```
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 0);

	std::string input_folder = "d:\\samples\\input";//or image file: input.jpg...
	std::string output_folder = "d:\\samples\\output";//or PDF file: output.pdf...
	std::string set_password = "secret";
	std::string log_file = "d:\\samples\\output\\image2pdf.log";

	//set thread number.
	int thread_number = 4;

	// log level.
	int log_level = 4;

	// the resolution (DPI:= Dots Pet Inch) for the output PDF file. Valid only when width
	// and height is not set.
	int dpi = 96;

	// recursion depth of search sub-folders. 0: search the full folders.
	int depth = 0;

	// page size of the output PDF file.
	int width = 500, height = 500;

	// page margin of the output PDF file. No margin by default.
	int margin_top = 20, margin_bottom = 20, margin_left = 20, margin_right = 20;

	// create bookmark for the output PDF file using image name.
	bool bookmark = true;

	// set title of PDF files.
	std::string title = "Foxit PDF Toolkit User Manual";

	// set subject of PDF files.
	std::string subject = "Foxit PDF Toolkit";

	// set keywords of PDF files.
	std::string keywords = "PDF Toolkit";

	// set author of PDF files.
	std::string author = "Jessie";

	// set file creation application information of PDF files.
	std::string creator = "Foxit Reader";

	// ------------------------------------------------
	// Given the above settings build a command string
	std::string strCommandline = "";

	if(!input_folder.empty())
```

```
		strCommandline.append("-i ").append(input_folder).append(" ");

	if(!output_folder.empty())
		strCommandline.append("-o " ).append(output_folder).append(" ");

	if(!set_password.empty())
		strCommandline.append("-sp ").append(set_password).append(" ");

	if(!title.empty())
		strCommandline.append("-title " ).append(title).append(" ");

	if(!subject.empty())
		strCommandline.append("-subject " ).append(subject).append(" ");

	if(!keywords.empty())
		strCommandline.append("-keywords " ).append(keywords).append(" ");

	if(!author.empty())
		strCommandline.append("-author " ).append(author).append(" ");

	if(!creator.empty())
		strCommandline.append("-creator " ).append(creator).append(" ");

	if(!log_file.empty())
		strCommandline.append("-log ").append(log_file).append(" ");

	const int MAX_LEGHT = 128;
	const int DATA_RADIX = 10;
	char temp[MAX_LEGHT] = {0};

	if (log_level > 0)
	{
		itoa(log_level,temp,DATA_RADIX);
		strCommandline.append("-l ").append(temp).append(" ");
	}

	if (width > 0 && height > 0)
	{
		memset(temp, 0 , MAX_LEGHT);
		itoa(width, temp ,DATA_RADIX);
		strCommandline.append("-width ").append(temp).append(" ");

		memset(temp, 0 , MAX_LEGHT);
		itoa(height, temp ,DATA_RADIX);
		strCommandline.append("-height ").append(temp).append(" ");
	}
	else
	{	//use dpi to set page size.
		if (dpi >=0 )
		{
			memset(temp, 0 ,MAX_LEGHT);
			itoa(dpi, temp ,DATA_RADIX);
			strCommandline.append("-dpi ").append(temp).append(" ");
		}
	}

	if (depth >= 0)
	{
		memset(temp, 0 , MAX_LEGHT);
		itoa(depth, temp ,DATA_RADIX);
```

```
            strCommandline.append("-depth ").append(temp).append(" ");
        }

        if (thread_number>= 0)
        {
            memset(temp, 0, MAX_LEGHT);
            itoa(thread_number, temp ,DATA_RADIX);
            strCommandline.append("-t ").append(temp).append(" ");
        }

        if (margin_top >= 0 || margin_right >= 0 || margin_bottom >= 0 || margin_left >= 0)
        {
            bool flag = false;
            if (margin_left >= 0)
            {
                memset(temp, 0 , MAX_LEGHT);
                itoa(margin_left, temp ,DATA_RADIX);
                strCommandline.append("-margin ").append(temp).append(" ");
                if(margin_top >= 0)
                {
                    memset(temp, 0 , MAX_LEGHT);
                    itoa(margin_top, temp ,DATA_RADIX);
                    strCommandline.append(temp).append(" ");
                    if(margin_right >= 0)
                    {
                        memset(temp, 0 , MAX_LEGHT);
                        itoa(margin_right, temp ,DATA_RADIX);
                        strCommandline.append(temp).append(" ");
                        if(margin_bottom >= 0)
                        {
                            memset(temp, 0 , MAX_LEGHT);
                            itoa(margin_bottom, temp ,DATA_RADIX);
                            strCommandline.append(temp).append(" ");
                        }
                    }
                }
            }
        }

        if(bookmark)   strCommandline.append("-b").append(" ");

        FXT_Image2PDFRun(strCommandline.c_str(), myCallBack, NULL);

        FXT_DestroyLibrary() ;
}
```

5.1.2 Reporting Progress Messages and Errors

To find out if Image2PDF processing was successful, the application can query the status code returned by FXT_Image2PDFRun().

For example,

```
int ret = FXT_Image2PDFRun(...);
if (ret == FXT_ERROR_SUCCESS) {
        // No errors...
}
```

```
else if (ret == FXT_ERROR_LICENSE) {
	// Invalid license...
}
else if (ret == FXT_ERROR_PARAM) {
	// Invalid param
}
else if (ret == FXT_ERROR_LIBRARY) {
	// Failed to initialize GSDK Library
}
else if (ret == FXT_ERROR_ERROR) {
	// Failed to convert image into pdf.
}
else {
	// Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error codes in the "include/fxpdftools.h" header.

For more detailed error and message reporting, you can pass a pointer to the custom callback function in the second parameter of FXT_Image2PDFRun(). The last parameter in FXT_Image2PDFRun() is a pointer to custom data that you may want to pass to the callback function.

The following is a sample callback function:

```
// Using C/C++
char* MyCallback(void* userData, int mode, char* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		cout << "Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		cout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		cout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static string password;
		cin >> password;
		return (char*)password.c_str();
	}
	return 0;
}
```

5.2 Office2PDF

5.2.1 Working with Office2PDF API

The following is the simplest application that can be built using Office2PDF API:

```
// Using C/C++
void main(int argc, char* argv[])
```

```
{
	int ret = FXT_InitLibrary("license_key", 0);
	FXT_ Office2PDFRun("-i d:\\input -o d:\\output", myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

This application converts all the Microsoft Office files in the “d:\input” folder into PDF files under the “d:\output” folder except for the secured files.

The command string (`"-i d:\\input -o d:\\output"`) of the above application is set by users directly in advance. Users also can get the command string through the command line. Building a command line application using Office2PDF API is as simple as the following:

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 0);

	std::string strCommandline = "";
	for(int i = 1; i < argc; i++)
		strCommandline.append(argv[i]).append(" ");
	FXT_ Office2PDFRun(strCommandline.c_str(), myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

It is also possible to build the command string dynamically (e.g. based on the user or a dynamic input) as illustrated in the following code snippet:

```
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 1);

	std::string input_folder = "d:\\samples\\input";//or office file: input.doc...
	std::string output_folder = "d:\\samples\\output";//or   PDF file: output.pdf...
	std::string open_password = "secret";
	std::string log_file = "d:\\samples\\output\\office2pdf.log";

	//set thread number
	int thread_number = 4;

	// log level
	int log_level = 4;


	// create bookmark for the output PDF file using headings of a Microsoft Word file.
	int bookmark = 1;

	// specify that the output PDF file(s) should be compliant with the PDF/A standard.
	bool pdfa = true;

	// specify the conversion mode for Microsoft Excel files.
	int scale = 3;
```

```
	// ------------------------------------------------
	// Given the above settings build a command string.
	std::string strCommandline = "";

	if(!input_folder.empty())
		strCommandline.append("-i ").append(input_folder).append(" ");

	if(!output_folder.empty())
		strCommandline.append("-o " ).append(output_folder).append(" ");

	if(!open_password.empty())
		strCommandline.append("-op ").append(open_password).append(" ");

	if(!log_file.empty())
		strCommandline.append("-log ").append(log_file).append(" ");

	const int MAX_LEGHT = 128;
	const int DATA_RADIX = 10;
	char temp[MAX_LEGHT] = {0};

	if(log_level > 0)
	{
		itoa(log_level,temp,DATA_RADIX);
		strCommandline.append("-l ").append(temp).append(" ");
	}

	if(thread_number>= 0)
	{
		memset(temp, 0 , MAX_LEGHT);
		itoa(thread_number, temp ,DATA_RADIX);
		strCommandline.append("-t ").append(temp).append(" ");
	}

	if(bookmark >= 0)
	{
		memset(temp, 0 , MAX_LEGHT);
		itoa(bookmark, temp ,DATA_RADIX);
		strCommandline.append("-b ").append(temp).append(" ");
	}

	if(pdfa)	strCommandline.append("-pdfa").append(" ");

	if(scale >= 0)
	{
		memset(temp, 0 , MAX_LEGHT);
		itoa(scale, temp ,DATA_RADIX);
		strCommandline.append("-scale ").append(temp).append(" ");
	}

	FXT_Office2PDFRun(strCommandline.c_str(), myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

5.2.2 Reporting Progress Messages and Errors

To find out if Office2PDF processing was successful, the application can query the status code returned by FXT_Office2PDFRun().

For example,

```
int ret = FXT_Office2PDFRun(...);
if (ret == FXT_ERROR_SUCCESS) {
	// No errors...
}
else if (ret == FXT_ERROR_LICENSE) {
	// Invalid license...
}
else if (ret == FXT_ERROR_PARAM) {
	// Invalid param
}
else if (ret == FXT_ERROR_LIBRARY) {
	// Failed to initialize GSDK Library
}
else {
	// Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error codes in the "include/fxpdftools.h" header.

For more detailed error and message reporting, you can pass a pointer to the custom callback function in the second parameter of FXT_Office2PDFRun(). The last parameter in FXT_Office2PDFRun() is a pointer to custom data that you may want to pass to the callback function.

The following is a sample callback function:

```
// Using C/C++
char* MyCallback(void* userData, int mode, char* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		cout << "Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		cout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		cout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static string password;
		cin >> password;
		return (char*)password.c_str();
	}
	return 0;
}
```

5.3 PDFWatermark

5.3.1 Working with PDFWatermark API

The following is the simplest application that can be built using PDFWatermark API:

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 0);
	int FXT_WatermarkRun("-i d:\\input -o d:\\output –conf d:\\conf_wm.xml", myCallBack,
NULL);

	FXT_DestroyLibrary();
}
```

This application adds a watermark into PDF files. All the PDF files in the “d:\input” folder will be added a watermark and output to “d:\output” folder except for the secured files.

The command string (`"-i d:\\input -o d:\\output –conf d:\\conf_wm.xml "`) of the above application is set by users directly in advance. Users also can get the command string through the command line. Building a command line application using PDFWatermark API is as simple as the following:

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 0);

	std::string strCommandline = "";
	for(int i = 1; i < argc; i++)
		strCommandline.append(argv[i]).append(" ");
	FXT_WatermarkRun(strCommandline.c_str(), myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

It is also possible to build the command string dynamically (e.g. based on the user or a dynamic input) as illustrated in the following code snippet:

```
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 0);

	std::string input_folder = "d:\\samples\\input";//or PDF file: input.pdf...
	std::string output_folder = "d:\\samples\\output";//or PDF file: output.pdf...
	std::string set_password = "secret";
	std::string log_file = "d:\\samples\\output\\watermark.log";
	std::string conf_file = "d:\\samples\\conf\\conf_wm.xml";//the configuration file
	// set title of PDF files.
	std::string title = "Foxit PDF Toolkit User Manual";
```

```
	// set subject of PDF files.
	std::string subject = "Foxit PDF Toolkit";

	// set keywords of PDF files.
	std::string keywords = "PDF Toolkit";

	// set author of PDF files.
	std::string author = "Jessie";

	// set file creation application information of PDF files.
	std::string creator = "Foxit Reader";

	//set thread number
	int thread_number = 4;

	// log level
	int log_level = 4;

	// ------------------------------------------------
	// Given the above settings build a command string.
	std::string strCommandline = "";

	if(!input_folder.empty())
		strCommandline.append("-i ").append(input_folder).append(" ");

	if(!output_folder.empty())
		strCommandline.append("-o " ).append(output_folder).append(" ");

	if(!conf_file.empty())
		strCommandline.append("-conf " ).append(conf_file).append(" ");

	if(!set_password.empty())
		strCommandline.append("-op ").append(set_password).append(" ");

	if(!title.empty())
		strCommandline.append("-title " ).append(title).append(" ");

	if(!subject.empty())
		strCommandline.append("-subject " ).append(subject).append(" ");

	if(!keywords.empty())
		strCommandline.append("-keywords " ).append(keywords).append(" ");

	if(!author.empty())
		strCommandline.append("-author " ).append(author).append(" ");

	if(!creator.empty())
		strCommandline.append("-creator " ).append(creator).append(" ");

	if(!log_file.empty())
		strCommandline.append("-log ").append(log_file).append(" ");

	if (log_level > 0)
	{
		itoa(log_level,temp,DATA_RADIX);
		strCommandline.append("-l ").append(temp).append(" ");
	}

	if (thread_number >= 0)
	{
```

```
            memset(temp, 0 , MAX_LEGHT);
            itoa(thread_number, temp ,DATA_RADIX);
            strCommandline.append("-t ").append(temp).append(" ");
      }

      FXT_WatermarkRun (strCommandline.c_str(), myCallBack, NULL);

      FXT_DestroyLibrary ();
```

5.3.2 Reporting Progress Messages and Errors

To find out if PDFWatermark processing was successful, the application can query the status code returned by FXT_WatermarkRun().

For example,

```
int ret = FXT_ WatermarkRun(...);
if (ret == FXT_ERROR_SUCCESS) {
      // No errors...
}
else if (ret == FXT_ERROR_LICENSE) {
      // Invalid license...
}
else if (ret == FXT_ERROR_PARAM) {
      // Invalid param
}
else if (ret == FXT_ERROR_LIBRARY) {
      // Failed to initialize GSDK Library
}
else if (ret == FXT_ERROR_ERROR) {
      // Failed to add a watermark to PDF file.
}
else {
      // Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error codes in the “include/fxpdftools.h” header.

For more detailed error and message reporting, you can pass a pointer to the custom callback function in the second parameter of FXT_WatermarkRun(). The last parameter in FXT_WatermarkRun() is a pointer to custom data that you may want to pass to the callback function.

The following is a sample callback function:

```
// Using C/C++
char* MyCallback(void* userData, int mode, char* msg, bool* isStop) {
      if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
            cout << "Error: " << msg << endl;
      }
      else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
```

```
                cout << msg;
        }
        else if (mode == CALLBACK_PDFTOOL_MSG) {
                cout << msg;
        }
        else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
                static string password;
                cin >> password;
                return (char*)password.c_str();
        }
        return 0;
}
```

5.4 PDFHeaderFooter

5.4.1 Working with PDFHeaderFooter API

The following is the simplest application that can be built using PDFHeaderFooter API:

```
// Using C/C++
void main(int argc, char* argv[])
{
        int ret = FXT_InitLibrary("license_key", 0);
        FXT_HeaderFooterRun("-i d:\\input -o d:\\output\\output.pdf –mode 1 –overlay –conf
d:\\conf_hf.xml", myCallBack, NULL);

        FXT_DestroyLibrary();
}
```

This application adds headers/footers into PDF files. All the PDF files in the “d:\input” folder will be added a header/footer and output to “d:\output” folder except for the secured files.

The command string (`"-i d:\\input -o d:\\output\\output.pdf –mode 1 –overlay –conf d:\\conf_hf.xml"`) of the above application is set by users directly in advance. Users also can get the command string through the command line. Building a command line application using PDFHeaderFooter API is as simple as the following:

```
// Using C/C++
void main(int argc, char* argv[])
{
        int ret = FXT_InitLibrary("license_key", 0);

        std::string strCommandline = "";
        for(int i = 1; i < argc; i++)
                strCommandline.append(argv[i]).append(" ");
        FXT_HeaderFooterRun(strCommandline.c_str(), myCallBack, NULL);

        FXT_DestroyLibrary();
}
```

It is also possible to build the command string dynamically (e.g. based on the user or a dynamic input) as illustrated in the following code snippet:

```
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 1);

	std::string input_folder = "d:\\samples\\input";//or PDF file: input.pdf...
	std::string output_folder = "d:\\samples\\output";//or   PDF file: output.pdf...
	std::string open_password = "secret";
	std::string log_file = "d:\\samples\\output\\headerfooter.log";
	std::string conf_file= "d:\\samples\\conf\\conf_hf.xml";// the configuration tool

	// set thread number
	int thread_number = 4;

	// log level
	int log_level = 4;

	// specify the mode to be used. Adds a new header/footer.
	int mode = 1;
	// set title of PDF files.
	std::string title = "Foxit PDF Toolkit User Manual";

	// set subject of PDF files.
	std::string subject = "Foxit PDF Toolkit";

	// set keywords of PDF files.
	std::string keywords = "PDF Toolkit";

	// set author of PDF files.
	std::string author = "Jessie";

	// set file creation application information of PDF files.
	std::string creator = "Foxit Reader";

	// ------------------------------------------------
	// Given the above settings build a command string.
	std::string strCommandline = "";

	if(!input_folder.empty())
		strCommandline.append("-i ").append(input_folder).append(" ");

	if(!output_folder.empty())
		strCommandline.append("-o ").append(output_folder).append(" ");

	if(!open_password.empty())
		strCommandline.append("-op ").append(open_password).append(" ");

	if(!title.empty())
		strCommandline.append("-title " ).append(title).append(" ");

	if(!subject.empty())
		strCommandline.append("-subject " ).append(subject).append(" ");

	if(!keywords.empty())
		strCommandline.append("-keywords " ).append(keywords).append(" ");
```

```
	if(!author.empty())
		strCommandline.append("-author " ).append(author).append(" ");

	if(!creator.empty())
		strCommandline.append("-creator " ).append(creator).append(" ");

	if(!log_file.empty())
		strCommandline.append("-log ").append(log_file).append(" ");

	const int MAX_LEGHT = 128;
	const int DATA_RADIX = 10;
	char temp[MAX_LEGHT] = {0};

	if (log_level > 0)
	{
		itoa(log_level,temp,DATA_RADIX);
		strCommandline.append("-l ").append(temp).append(" ");
	}

	if (thread_number>= 0)
	{
		memset(temp, 0 , MAX_LEGHT);
		itoa(thread_number, temp ,DATA_RADIX);
		strCommandline.append("-t ").append(temp).append(" ");
	}

	if (mode >= 0)
	{
		memset(temp, 0 , MAX_LEGHT);
		itoa(mode, temp ,DATA_RADIX);
		strCommandline.append("-mode ").append(temp).append(" ");
	}

	if(!config.empty())
		strCommandline.append("-conf ").append(conf_file).append(" ");

	FXT_HeaderFooterRun(strCommandline.c_str(), myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

5.4.2 Reporting Progress Messages and Errors

To find out if PDFHeaderFooter processing was successful, the application can query the status code returned by FXT_HeaderFooterRun().

For example,

```
int ret = FXT_HeaderFooterRun(...);
if (ret == FXT_ERROR_SUCCESS) {
	// No errors...
}
else if (ret == FXT_ERROR_LICENSE) {
	// Invalid license...
}
else if (ret == FXT_ERROR_PARAM) {
	// Invalid param
```

```
}
else if (ret == FXT_ERROR_LIBRARY) {
	// Failed to initialize GSDK Library
}
else if (ret == FXT_ERROR_ERROR) {
	// Failed to add header and footer to PDF file.
}
else {
	// Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error codes in the "include/fxpdftools.h" header.

For more detailed error and message reporting, you can pass a pointer to the custom callback function in the second parameter of FXT_Header0FooterRun(). The last parameter in FXT_HeaderFooterRun() is a pointer to custom data that you may want to pass to the callback function.

The following is a sample callback function:

```
// Using C/C++
char* MyCallback(void* userData, int mode, char* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		cout << "Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		cout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		cout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static string password;
		cin >> password;
		return (char*)password.c_str();
	}
	return 0;
}
```

5.5 PDFOptimizer

5.5.1 Working with PDFOptimizer API

The following is the simplest application that can be built using PDFOptimizer API:

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 0);
	FXT_OptimizerRun("-i d:\\input -o d:\\output", myCallBack, NULL);

	FXT_DestroyLibrary();
```

```
}
```

This application optimizes PDF files. All the PDF files in the “d:\input” folder will be optimized and output to “d:\output” folder except for the secured files.

The command string (`"-i d:\\input -o d:\\output\\output.pdf"`) of the above application is set by users directly in advance. Users also can get the command string through the command line. Building a command line application using Image2PDF API is as simple as the following:

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 0);

	std::string strCommandline = "";
	for(int i = 1; i < argc; i++)
		strCommandline.append(argv[i]).append(" ");
	FXT_OptimizerRun(strCommandline.c_str(), myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

It is also possible to build the command string dynamically (e.g. based on the user or a dynamic input) as illustrated in the following code snippet:

```
int main(int argc, char* argv[])
{
	int ret =  FXT_InitLibrary("license_key", 0);

	std::string input_folder = "d:\\samples\\input";   //or PDF file: input.pdf
	std::string output_folder = "d:\\samples\\output"; //or PDF file: output.pdf
	std::string set_password = "secret";
	std::string log_file = "d:\\samples\\Output\\pdfoptimizer.log";

	// set title of PDF files.
	std::string title = "Foxit PDF Toolkit User Manual";

	// set subject of PDF files.
	std::string subject = "Foxit PDF Toolkit";

	// set keywords of PDF files.
	std::string keywords = "PDF Toolkit";

	// set author of PDF files.
	std::string author = "Jessie";

	// set file creation application information of PDF files.
	std::string creator = "Foxit Reader";

	//set thread number.
	int thread_number = 4;

	// log level
	int log_level = 4;
```

```
// recursion depth of search sub-folders. 0: search the full folders.
int depth = 0;

// ------------------------------------------------
// Given the above settings build a command string.
std::string strCommandline = "";

if(!input_folder.empty())
	strCommandline.append("-i ").append(input_folder).append(" ");

if(!output_folder.empty())
	strCommandline.append("-o " ).append(output_folder).append(" ");

if(!set_password.empty())
	strCommandline.append("-op ").append(set_password).append(" ");

if(!title.empty())
	strCommandline.append("-title " ).append(title).append(" ");

if(!subject.empty())
	strCommandline.append("-subject " ).append(subject).append(" ");

if(!keywords.empty())
	strCommandline.append("-keywords " ).append(keywords).append(" ");

if(!author.empty())
	strCommandline.append("-author " ).append(author).append(" ");

if(!creator.empty())
	strCommandline.append("-creator " ).append(creator).append(" ");

if(!log_file.empty())
	strCommandline.append("-log ").append(log_file).append(" ");

const int MAX_LEGHT = 128;
const int DATA_RADIX  = 10;
char temp[MAX_LEGHT] = {0};

if (thread_number >= 0)		//set thread number
{
	memset(temp, 0 , MAX_LEGHT);
	itoa(thread_number, temp ,DATA_RADIX);
	strCommandline.append("-t ").append(temp).append(" ");
}

if (log_level > 0)
{
	itoa(log_level,temp,DATA_RADIX);
	strCommandline.append("-l ").append(temp).append(" ");
}

if (depth >= 0)
{
	memset(temp, 0 , MAX_LEGHT);
	itoa(depth, temp ,DATA_RADIX);
	strCommandline.append("-r ").append(temp).append(" ");
}

//set color/gray images DownSample
```

```
        std::string dcAlgorithm = "b";  //or "a","s"
        std::string dcDpiAbove  = "225";
        std::string dcDpiSet    = "150";
        strCommandline.append("-dc ").append(dcAlgorithm).append(" ");
        strCommandline.append(dcDpiAbove).append(" ");
        strCommandline.append(dcDpiSet).append(" ");

        //set color/gray images compress
        std::string ccAlgorithm = "j";  //or "j2"
        std::string ccLevel = "medium";  //or "min","low","high","max"
        strCommandline.append("-cc ").append(ccAlgorithm).append(" ");
        strCommandline.append(ccLevel).append(" ");

        //set monochrome images DownSample
        std::string dmAlgorithm = "b";    //or "a","s"
        std::string dmDpiAbove  = "450";
        std::string dmDpiSet    = "300";
        strCommandline.append("-dm ").append(dmAlgorithm).append(" ");
        strCommandline.append(dmDpiAbove).append(" ");
        strCommandline.append(dmDpiSet).append(" ");

        //set monochrome images compress
        std::string cmAlgorithm = "jbig2";  //or "ccitt","runlength"
        std::string cmLevel = "lossless";   //or "lossy"
        strCommandline.append("-cm ").append(cmAlgorithm).append(" ");
        strCommandline.append(cmLevel).append(" ");

        //Discard objects or User Data.
        std::string discardList = "\"1-11\"";
        strCommandline.append("-d ").append(discardList).append(" ");

        //Cleans up streams, bookmarks, or links.
        std::string cleanupList = "\"1-4\"";
        strCommandline.append("-cl ").append(cleanupList).append(" ");

        strCommandline.append("-rd ");

        //Unembeds all fonts in selected PDF document(s).
        strCommandline.append("-u ");

        FXT_OptimizerRun(strCommandline.c_str(), myCallBack, NULL);
        FXT_DestroyLibrary();
}
```

5.5.2 Reporting Progress Messages and Errors

To find out if PDFOptimizer processing was successful, the application can query the status code returned by FXT_OptimizerRun().

For example,

```
int ret = FXT_OptimizerRun(...);
if (ret == FXT_ERROR_SUCCESS) {
        // No errors...
}
else if (ret == FXT_ERROR_LICENSE) {
        // Invalid license...
```

```
}
else if (ret == FXT_ERROR_PARAM) {
	// Invalid parameter
}
else if (ret == FXT_ERROR_LIBRARY) {
	// Failed to initialize GSDK Library
}
else if (ret == FXT_ERROR_ERROR) {
	// Failed to optimize PDF file
}
else {
	// Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error codes in the "include/fxpdftools.h" header.

For more detailed error and message reporting, you can pass a pointer to the custom callback function in the second parameter of FXT_OptimizerRun(). The last parameter in FXT_OptimizerRun() is a pointer to custom data that you may want to pass to the callback function.

The following is a sample callback function:

```
// Using C/C++
char* MyCallback(void* userData, int mode, char* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		cout << "Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		cout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		cout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static string password;
		cin >> password;
		return (char*)password.c_str();
	}
	return 0;
}
```

5.6 PDFRedactor

5.6.1 Working with PDFRedactor API

The following is the simplest application that can be built using PDFRedactor API:

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 0);
```

```
	FXT_RedactorRun("-i d:\\input -o d:\\output -mode 1 -keyword "foxit"", myCallBack,
NULL);

	FXT_DestroyLibrary();
}
```

This application redacts (removes) the sensitive content “Foxit” from the PDF files in the “d:\input” folder except for the secured files, and the redacted PDF files will be saved to the “d:\output” folder.

The command string (`"-i d:\\input -o d:\\output -mode 1 -keyword "foxit""`) of the above application is set by users directly in advance. Users also can get the command string through command line. Building a command line application using PDFRedactor API is as simple as the following:

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 0);

	std::string strCommandline = "";
	for(int i = 1; i < argc; i++)
		strCommandline.append(argv[i]).append(" ");
	FXT_RedactorRun(strCommandline.c_str(), myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

It is also possible to build the command string dynamically (e.g. based on the user or a dynamic input) as illustrated in the following code snippet:

```
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 0);

	std::string input_folder = "d:\\samples\\input";//or a PDF file: input.pdf
	std::string output_folder = "d:\\samples\\output";//or a PDF file: output.pdf...
	std::string open_password = "secret"; //open password
	std::string log_file = "d:\\samples\\output\\convert2pdf.log";

	// the mode of keyword to be searched.
	int mode = 1;

	// the keyword needed to be searched.
	std::string keyword = "Foxit";

	// the characters of the keyword you want to redact.
	int character = 3;

	// the page range to apply redaction.
	std::string range = "1,5,9";

	// fill color for redaction marks.
	int markcolor = 255 255 0;

	// specify the overlay text for redaction marks.
```

```
	std::string tx = "secret";

	// the alignment of the overlay text.
	int ta = 1;

	// the font style of the overlay text.
	std::string font = "Calibri";

	// the font size of the overlay text.
	int fs = 11;

	// font color of the overlay text.
	int fontcolor = 200 255 0;

	// Search the specified keyword in both PDF forms and the main body of text.
	bool form = true;

	// set title of PDF files.
	std::string title = "Foxit PDF Toolkit User Manual";

	// set subject of PDF files.
	std::string subject = "Foxit PDF Toolkit";

	// set keywords of PDF files.
	std::string keywords = "PDF Toolkit";

	// set author of PDF files.
	std::string author = "Jessie";

	// set file creation application information of PDF files.
	std::string creator = "Foxit Reader";

	// recursion depth of search sub-folders. 0: search all of the full folders.
	int depth = 0;

	//set thread number.
	int thread_number = 4;

	// log level.
	int log_level = 4;

	// ------------------------------------------------
	// Given the above settings build a command string.
	std::string strCommandline = "";

	if(!input_folder.empty())
		strCommandline.append("-i ").append(input_folder).append(" ");

	if(!output_folder.empty())
		strCommandline.append("-o " ).append(output_folder).append(" ");

	if(!keyword.empty())
		strCommandline.append("-keyword ").append(keyword).append(" ");

	if(!range.empty())
		strCommandline.append("-range ").append(range).append(" ");

	if(!tx.empty())
		strCommandline.append("-tx ").append(tx).append(" ");
```

```
	if(!font.empty())
		strCommandline.append("-font " ).append(font).append(" ");

	if(!open_password.empty())
		strCommandline.append("-op ").append(open_password).append(" ");

	if(!title.empty())
		strCommandline.append("-title " ).append(title).append(" ");

	if(!subject.empty())
		strCommandline.append("-subject " ).append(subject).append(" ");

	if(!keywords.empty())
		strCommandline.append("-keywords " ).append(keywords).append(" ");

	if(!author.empty())
		strCommandline.append("-author " ).append(author).append(" ");

	if(!creator.empty())
		strCommandline.append("-creator " ).append(creator).append(" ");

	if(!log_file.empty())
		strCommandline.append("-log ").append(log_file).append(" ");

	const int MAX_LEGHT = 128;
	const int DATA_RADIX = 10;
	char temp[MAX_LEGHT] = {0};

	if (log_level > 0)
	{
		itoa(log_level,temp,DATA_RADIX);
		strCommandline.append("-l ").append(temp).append(" ");
	}

	if (depth >= 0)
	{
		memset(temp, 0 , MAX_LEGHT);
		itoa(depth, temp ,DATA_RADIX);
		strCommandline.append("-depth ").append(temp).append(" ");
	}

	if (thread_number>= 0)
	{
		memset(temp, 0, MAX_LEGHT);
		itoa(thread_number, temp ,DATA_RADIX);
		strCommandline.append("-t ").append(temp).append(" ");
	}

	if (mode >= 0)
	{
		memset(temp, 0, MAX_LEGHT);
		itoa(mode, temp ,DATA_RADIX);
		strCommandline.append("-mode ").append(temp).append(" ");
	}

	if (character >= 0)
	{
		memset(temp, 0, MAX_LEGHT);
		itoa(character, temp ,DATA_RADIX);
		strCommandline.append("-character ").append(temp).append(" ");
```

```
	}

	if (R >= 0 && G >= 0 && B >= 0)
	{
		memset(temp, 0, MAX_LEGHT);
		itoa(markcolor, temp ,DATA_RADIX);
		strCommandline.append("-markcolor ").append(temp).append(" ");
	}

	if (ta>= 0)
	{
		memset(temp, 0, MAX_LEGHT);
		itoa(ta, temp ,DATA_RADIX);
		strCommandline.append("-ta ").append(temp).append(" ");
	}

	if (fs>= 0)
	{
		memset(temp, 0, MAX_LEGHT);
		itoa(fs, temp ,DATA_RADIX);
		strCommandline.append("-fs ").append(temp).append(" ");
	}

	if (R >= 0 && G >= 0 && B >= 0)
	{
		memset(temp, 0, MAX_LEGHT);
		itoa(fontcolor, temp ,DATA_RADIX);
		strCommandline.append("-fontcolor ").append(temp).append(" ");
	}

	if(form)	strCommandline.append("-form").append(" ");

	FXT_RedactorRun(strCommandline.c_str(), myCallBack, NULL);

	FXT_DestroyLibrary() ;
}
```

5.6.2 Reporting Progress Messages and Errors

To find out if PDFRedactor processing was successful, the application can query the status code returned by FXT_RedactorRun().

For example,

```
int ret = FXT_RedactorRun(...);
if (ret == FXT_ERROR_SUCCESS) {
	// No errors...
}
else if (ret == FXT_ERROR_LICENSE) {
	// Invalid license...
}
else if (ret == FXT_ERROR_PARAM) {
	// Invalid param
}
else if (ret == FXT_ERROR_LIBRARY) {
	// Failed to initialize GSDK Library
}
```

```
else if (ret == FXT_ERROR_ERROR) {
	// Failed to redact PDF file.
}
else {
	// Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error code in the "include/fxpdftools.h" header.

For more detailed error and message reporting, you can pass a pointer to the custom callback function in the second parameter of FXT_RedactorRun(). The last parameter in FXT_RedactorRun() is a pointer to custom data that you may want to pass to the callback function.

The following is a sample callback function:

```
// Using C/C++
char* MyCallback(void* userData, int mode, char* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		cout << "Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		cout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		cout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static string password;
		cin >> password;
		return (char*)password.c_str();
	}
	return 0;
}
```

5.7 PDFMetadata

5.7.1 Working with PDFMetadata API

The following is the simplest application that can be built using PDFMetadata API:

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 0);
	FXT_MetadataRun("-i d:\\input -o d:\\output -title "Foxit PDF Toolkit User Manual"",
myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

This application adds document title information to the PDF files in the “d:\input” folder except for the secured files, and the output PDF files will be saved to the “d:\output” folder.

The command string (`"-i d:\\input -o d:\\output -title "Foxit PDF Toolkit User Manual""`) of the above application is set by users directly in advance. Users also can get the command string through command line. Building a command line application using PDFMetadata API is as simple as the following:

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 0);

	std::string strCommandline = "";
	for(int i = 1; i < argc; i++)
		strCommandline.append(argv[i]).append(" ");
	FXT_MetadataRun(strCommandline.c_str(), myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

It is also possible to build the command string dynamically (e.g. based on the user or a dynamic input) as illustrated in the following code snippet:

```
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 0);

	std::string input_folder = "d:\\samples\\input";//or a PDF file: input.pdf
	std::string output_folder = "d:\\samples\\output";//or a PDF file: output.pdf...
	std::string open_password = "secret"; //open password
	std::string log_file = "d:\\samples\\output\\pdfmetadata.log";

	// set title of PDF files.
	std::string title = "Foxit PDF Toolkit User Manual";

	// set subject of PDF files.
	std::string subject = "Foxit PDF Toolkit";

	// set keywords of PDF files.
	std::string keywords = "PDF Toolkit";

	// set author of PDF files.
	std::string author = "Jessie";

	// set file creation application information of PDF files.
	std::string creator = "Foxit Reader";

	// recursion depth of search sub-folders. 0: search all of the full folders.
	int depth = 0;

	//set thread number.
	int thread_number = 4;

	// log level.
```

```
int log_level = 4;

// ------------------------------------------------
// Given the above settings build a command string.
std::string strCommandline = "";

if(!input_folder.empty())
	strCommandline.append("-i ").append(input_folder).append(" ");

if(!output_folder.empty())
	strCommandline.append("-o " ).append(output_folder).append(" ");

if(!title.empty())
	strCommandline.append("-title " ).append(title).append(" ");

if(!subject.empty())
	strCommandline.append("-subject " ).append(subject).append(" ");

if(!keywords.empty())
	strCommandline.append("-keywords " ).append(keywords).append(" ");

if(!author.empty())
	strCommandline.append("-author " ).append(author).append(" ");

if(!creator.empty())
	strCommandline.append("-creator " ).append(creator).append(" ");

if(!open_password.empty())
	strCommandline.append("-op ").append(open_password).append(" ");

if(!log_file.empty())
	strCommandline.append("-log ").append(log_file).append(" ");

const int MAX_LEGHT = 128;
const int DATA_RADIX = 10;
char temp[MAX_LEGHT] = {0};

if (log_level > 0)
{
	itoa(log_level,temp,DATA_RADIX);
	strCommandline.append("-l ").append(temp).append(" ");
}

if (depth >= 0)
{
	memset(temp, 0 , MAX_LEGHT);
	itoa(depth, temp ,DATA_RADIX);
	strCommandline.append("-depth ").append(temp).append(" ");
}

if (thread_number>= 0)
{
	memset(temp, 0, MAX_LEGHT);
	itoa(thread_number, temp ,DATA_RADIX);
	strCommandline.append("-t ").append(temp).append(" ");
}

FXT_MetadataRun(strCommandline.c_str(), myCallBack, NULL);

FXT_DestroyLibrary() ;
```

```
}
```

5.7.2 Reporting Progress Messages and Errors

To find out if PDFMetadata processing was successful, the application can query the status code returned by FXT_MetadataRun().

For example,

```
int ret = FXT_MetadataRun(...);
if (ret == FXT_ERROR_SUCCESS) {
	// No errors...
}
else if (ret == FXT_ERROR_LICENSE) {
	// Invalid license...
}
else if (ret == FXT_ERROR_PARAM) {
	// Invalid param
}
else if (ret == FXT_ERROR_LIBRARY) {
	// Failed to initialize GSDK Library
}
else if (ret == FXT_ERROR_ERROR) {
	// Failed to set metadata information of PDF file
}
else {
	// Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error code in the "include/fxpdftools.h" header.

For more detailed error and message reporting, you can pass a pointer to the custom callback function in the second parameter of FXT_MetadataRun(). The last parameter in FXT_MetadataRun() is a pointer to custom data that you may want to pass to the callback function.

The following is a sample callback function:

```
// Using C/C++
char* MyCallback(void* userData, int mode, char* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		cout << "Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		cout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		cout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static string password;
		cin >> password;
```

```
            return (char*)password.c_str();
        }
        return 0;
}
```

5.8 PDF2Text

5.8.1 Working with PDF2Text API

The following is the simplest application that can be built using PDF2Text API:

```
// Using C/C++
void main(int argc, char* argv[])
{
        int ret = FXT_InitLibrary("license_key", 0);
        FXT_PDF2TextRun("-i d:\\input -o d:\\output", myCallBack, NULL);

        FXT_DestroyLibrary();
}
```

This application converts all the PDF files in the “d:\input” folder into text files under the “d:\output” folder except for the secured files.

The command string (`"-i d:\\input -o d:\\output"`) of the above application is set by users directly in advance. Users also can get the command string through the command line. Building a command line application using PDF2Text API is as simple as the following:

```
// Using C/C++
void main(int argc, char* argv[])
{
        int ret = FXT_InitLibrary("license_key", 0);

        std::string strCommandline = "";
        for(int i = 1; i < argc; i++)
                strCommandline.append(argv[i]).append(" ");
        FXT_PDF2TextRun(strCommandline.c_str(), myCallBack, NULL);

        FXT_DestroyLibrary();
}
```

It is also possible to build the command string dynamically (e.g. based on the user or a dynamic input) as illustrated in the following code snippet:

```
void main(int argc, char* argv[])
{
        int ret = FXT_InitLibrary("license_key", 0);

        std::string input_folder = "d:\\samples\\input";//or a PDF file: input.pdf
        std::string output_folder = "d:\\samples\\output";//or a text file: output.txt...
        std::string open_password = "secret"; //open password
        std::string log_file = "d:\\samples\\output\\PDF2Text.log";
```

```
// the page range to convert.
std::string range = "all";

// get the total number of characters on each page.
bool charcount = false;

// print the number of characters to the screen.
bool printcount = false;

// ignore the PDF page layout.
bool notype = true;

// ignore the page break in the PDF.
bool nopagenumber = true;

// convert each PDF page into individual text files.
bool singlepage = true;

// Set Unicode text encoding to UTF16.
std::string encoding = "UTF16";

// recursion depth of search sub-folders. 0: search all of the full folders
int depth = 0;

//set thread number
int thread_number = 4;

// log level
int log_level = 4;

// ------------------------------------------------
// Given the above settings build a command string.
std::string strCommandline = "";

if(!input_folder.empty())
	strCommandline.append("-i ").append(input_folder).append(" ");

if(!output_folder.empty())
	strCommandline.append("-o " ).append(output_folder).append(" ");

if(!open_password.empty())
	strCommandline.append("-op ").append(open_password).append(" ");

if(!range.empty())
	strCommandline.append("-range " ).append(range).append(" ");

if(!encoding.empty())
	strCommandline.append("-encoding " ).append(encoding).append(" ");

if(!log_file.empty())
	strCommandline.append("-log ").append(log_file).append(" ");

const int MAX_LEGHT = 128;
const int DATA_RADIX = 10;
char temp[MAX_LEGHT] = {0};

if (log_level > 0)
{
	itoa(log_level,temp,DATA_RADIX);
```

```
            strCommandline.append("-l ").append(temp).append(" ");
        }

        if (depth >= 0)
        {
            memset(temp, 0 , MAX_LEGHT);
            itoa(depth, temp ,DATA_RADIX);
            strCommandline.append("-depth ").append(temp).append(" ");
        }

        if (thread_number>= 0)
        {
            memset(temp, 0, MAX_LEGHT);
            itoa(thread_number, temp ,DATA_RADIX);
            strCommandline.append("-t ").append(temp).append(" ");
        }

        if(charcount)   strCommandline.append("-charcount").append(" ");

        if(printcount)  strCommandline.append("-printcount").append(" ");

        if(notype)      strCommandline.append("-notype").append(" ");

        if(nopagenumber)      strCommandline.append("-nopagenumber").append(" ");

        if(singlepage)  strCommandline.append("-singlepage").append(" ");

        FXT_PDF2TextRun(strCommandline.c_str(), myCallBack, NULL);

        FXT_DestroyLibrary() ;
}
```

5.8.2 Reporting Progress Messages and Errors

To find out if PDF2Text processing was successful, the application can query the status code returned by FXT_PDF2TextRun().

For example,

```
int ret = FXT_PDF2TextRun(...);
if (ret == FXT_ERROR_SUCCESS) {
        // No errors...
}
else if (ret == FXT_ERROR_LICENSE) {
        // Invalid license...
}
else if (ret == FXT_ERROR_PARAM) {
        // Invalid param
}
else if (ret == FXT_ERROR_LIBRARY) {
        // Failed to initialize GSDK Library
}
else if (ret == FXT_ERROR_ERROR) {
        // Failed to convert PDF file to text file
}
else {
        // Other error
```

```
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error codes in the "include/fxpdftools.h" header.

For more detailed error and message reporting, you can pass a pointer to the custom callback function in the second parameter of FXT_PDF2TextRun(). The last parameter in FXT_PDF2TextRun() is a pointer to custom data that you may want to pass to the callback function.

The following is a sample callback function:

```
// Using C/C++
char* MyCallback(void* userData, int mode, char* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		cout << "Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		cout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		cout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static string password;
		cin >> password;
		return (char*)password.c_str();
	}
	return 0;
}
```

5.9 Text2PDF

5.9.1 Working with Text2PDF API

The following is the simplest application that can be built using Text2PDF API:

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 0);
	FXT_Text2PDFRun("-i d:\\input -o d:\\output", myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

This application converts all the text files in the "d:\input" folder into PDF files under the "d:\output" folder.

The command string ("-i d:\\input -o d:\\output") of the above application is set by users directly in advance. Users also can get the command string through the command line. Building a command line application using Text2PDF API is as simple as the following:

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 0);

	std::string strCommandline = "";
	for(int i = 1; i < argc; i++)
		strCommandline.append(argv[i]).append(" ");
	FXT_Text2PDFRun(strCommandline.c_str(), myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

It is also possible to build the command string dynamically (e.g. based on the user or a dynamic input) as illustrated in the following code snippet:

```
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary("license_key", 0);

	std::string input_folder = "d:\\samples\\input";//or a text file: input.txt
	std::string output_folder = "d:\\samples\\output";//or a PDF file: output.pdf...
	std::string set_password = "secret"; //open password for office files
	std::string log_file = "d:\\samples\\output\\convert2pdf.log";

	// page size of the output PDF file
	int width = 500, height = 500;

	// page margin of the output PDF file.
	int margin_top = 20, margin_bottom = 20, margin_left = 20, margin_right = 20;

	// font style of the output PDF file
	std::string font = "Calibri";

	// font size of the output PDF file
	int fs = 11;

	// font color of the output PDF file
	int fontcolor = 255 255 0;

	// set title of PDF files.
	std::string title = "Foxit PDF Toolkit User Manual";

	// set subject of PDF files.
	std::string subject = "Foxit PDF Toolkit";

	// set keywords of PDF files.
	std::string keywords = "PDF Toolkit";

	// set author of PDF files.
	std::string author = "Jessie";
```

```
	// set file creation application information of PDF files.
	std::string creator = "Foxit Reader";

	// recursion depth of search sub-folders. 0: search all of the full folders
	int depth = 0;

	//set thread number.
	int thread_number = 4;

	// log level.
	int log_level = 4;

	// ------------------------------------------------
	// Given the above settings build a command string.
	std::string strCommandline = "";

	if(!input_folder.empty())
		strCommandline.append("-i ").append(input_folder).append(" ");

	if(!output_folder.empty())
		strCommandline.append("-o " ).append(output_folder).append(" ");

	if(!set_password.empty())
		strCommandline.append("-sp ").append(set_password).append(" ");

	if(!font.empty())
		strCommandline.append("-font " ).append(font).append(" ");

	if(!title.empty())
		strCommandline.append("-title " ).append(title).append(" ");

	if(!subject.empty())
		strCommandline.append("-subject " ).append(subject).append(" ");

	if(!keywords.empty())
		strCommandline.append("-keywords " ).append(keywords).append(" ");

	if(!author.empty())
		strCommandline.append("-author " ).append(author).append(" ");

	if(!creator.empty())
		strCommandline.append("-creator " ).append(creator).append(" ");

	if(!log_file.empty())
		strCommandline.append("-log ").append(log_file).append(" ");

	const int MAX_LEGHT = 128;
	const int DATA_RADIX = 10;
	char temp[MAX_LEGHT] = {0};

	if (log_level > 0)
	{
		itoa(log_level,temp,DATA_RADIX);
		strCommandline.append("-l ").append(temp).append(" ");
	}

	if (depth >= 0)
	{
		memset(temp, 0 , MAX_LEGHT);
		itoa(depth, temp ,DATA_RADIX);
```

```
        strCommandline.append("-depth ").append(temp).append(" ");
    }

    if (thread_number>= 0)
    {
        memset(temp, 0, MAX_LEGHT);
        itoa(thread_number, temp ,DATA_RADIX);
        strCommandline.append("-t ").append(temp).append(" ");
    }

    if (width > 0 && height > 0)
    {
        memset(temp, 0 , MAX_LEGHT);
        itoa(width, temp ,DATA_RADIX);
        strCommandline.append("-width ").append(temp).append(" ");

        memset(temp, 0 , MAX_LEGHT);
        itoa(height, temp ,DATA_RADIX);
        strCommandline.append("-height ").append(temp).append(" ");
    }

    if (margin_top >= 0 || margin_right >= 0 || margin_bottom >= 0 || margin_left >= 0)
    {
        bool flag = false;
        if (margin_left >= 0)
        {
            memset(temp, 0 , MAX_LEGHT);
            itoa(margin_left, temp ,DATA_RADIX);
            strCommandline.append("-margin ").append(temp).append(" ");
            if(margin_top >= 0)
            {
                memset(temp, 0 , MAX_LEGHT);
                itoa(margin_top, temp ,DATA_RADIX);
                strCommandline.append(temp).append(" ");
                if(margin_right >= 0)
                {
                    memset(temp, 0 , MAX_LEGHT);
                    itoa(margin_right, temp ,DATA_RADIX);
                    strCommandline.append(temp).append(" ");
                    if(margin_bottom >= 0)
                    {
                        memset(temp, 0 , MAX_LEGHT);
                        itoa(margin_bottom, temp ,DATA_RADIX);
                        strCommandline.append(temp).append(" ");
                    }
                }
            }
        }
    }

    if (fs>= 0)
    {
        memset(temp, 0, MAX_LEGHT);
        itoa(fs, temp ,DATA_RADIX);
        strCommandline.append("-fs ").append(temp).append(" ");
    }

    if (R >= 0 && G >= 0 && B >= 0)
    {
        memset(temp, 0, MAX_LEGHT);
```

```
            itoa(fontcolor, temp ,DATA_RADIX);
            strCommandline.append("-fontcolor ").append(temp).append(" ");
        }

        FXT_Text2PDFRun(strCommandline.c_str(), myCallBack, NULL);

        FXT_DestroyLibrary() ;
}
```

5.9.2 Reporting Progress Messages and Errors

To find out if Text2PDF processing was successful, the application can query the status code returned by FXT_Text2PDFRun().

For example,

```
int ret = FXT_Text2PDFRun(...);
if (ret == FXT_ERROR_SUCCESS) {
        // No errors...
}
else if (ret == FXT_ERROR_LICENSE) {
        // Invalid license...
}
else if (ret == FXT_ERROR_PARAM) {
        // Invalid param
}
else if (ret == FXT_ERROR_LIBRARY) {
        // Failed to initialize GSDK Library
}
else if (ret == FXT_ERROR_ERROR) {
        // Failed to convert text file to PDF file
}
else {
        // Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error codes in the “include/fxpdftools.h” header.

For more detailed error and message reporting, you can pass a pointer to the custom callback function in the second parameter of FXT_Text2PDFRun(). The last parameter in FXT_Text2PDFRun() is a pointer to custom data that you may want to pass to the callback function.

The following is a sample callback function:

```
// Using C/C++
char* MyCallback(void* userData, int mode, char* msg, bool* isStop) {
        if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
                cout << "Error: " << msg << endl;
        }
        else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
                cout << msg;
```

```
        }
        else if (mode == CALLBACK_PDFTOOL_MSG) {
                cout << msg;
        }
        else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
                static string password;
                cin >> password;
                return (char*)password.c_str();
        }
        return 0;
}
```

6 Support

6.1 Reporting Problem

Should you encounter any technical questions or bug issues when using Foxit PDF Toolkit command line tools, please submit the problem report to Foxit support team at http://www.foxitsoftware.com/support/. In order to better help you solve the problem, please provide the following information:

- Contact details
- Product name and its version
- Your Operating System
- Detailed description of the problem
- Any other related information, such as error screenshot

Note *In the unfortunate event that Foxit PDF Toolkit should crash, Foxit PDF Toolkit will generate two files named "CRASHLOG.TXT" and "CRASH.DMP" under the current execution directory. The "CRASHLOG.TXT" file will record the detailed information of the module and the system that are used at the time the crash occurred. To help our engineering team track down the problem and provide a solution, please submit the two files to Foxit support team. Thank you for your cooperation.*

6.2 Contact Information

You can contact Foxit directly, please use the contact information as follows:

Foxit Support:

- http://www.foxitsoftware.com/support/

Sales Contact:

- Phone: 1-866-680-3668
- Email: sales@foxitsoftware.com

Support & General Contact:

- Phone: 1-866-MYFOXIT or 1-866-693-6948
- Email: support@foxitsoftware.com